Universidad Zaragoza
1542

Trabajo Fin de Grado

Diseño y desarrollo de un dispositivo de medición de CO2 con registro de datos a través de conexión inalámbrica.

Design and development of a CO2 measuring device with data records through wireless connection.

Autor

Fernando Gracia Alconchel

Directores

Juan Antonio Tejero Gómez

Pedro Abad Martín

Escuela de Ingeniería y Arquitectura
2021

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Fernando Gracia Alconchel ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO ING. TECNOLOGÍAS INDUSTRIALES (Título del Trabajo)

DISEÑO Y DESARROLLO DE UN DISPOSITIVO DE MEDICIÓN DE CO2 CON REGISTRO DE DATOS A TRAVÉS DE CONEXIÓN INALÁMBRICA

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 18 DE JUNIO DE 2021

Fdo: Fernando Gracia Alconchel

RESUMEN DEL PROYECTO

"Diseño y desarrollo de un dispositivo de medición de CO2 con registro de datos a través de conexión inalámbrica"

Realizado por: Fernando Gracia Alconchel

Director: Juan Antonio Tejero Gómez

Codirector: Pedro Abad Martín

En el presente proyecto se lleva a cabo el diseño y posterior desarrollo de un dispositivo para la medición de la concentración de dióxido de carbono en el ambiente, destinado específicamente a su uso en el ámbito académico.

Basándose en la oferta existente de dispositivos similares y las necesidades de uso actuales derivadas de la aparición del Covid-19, se establecen una serie de objetivos y capacidades que el dispositivo final deba alcanzar.

Marcados los objetivos de diseño del dispositivo, se lleva a cabo la elección de componentes de este, y se inicia el proceso de montaje, detallando paso a paso el funcionamiento de cada uno de los componentes y los resultados obtenidos.

Tras obtener un prototipo capaz de llevar a cabo las funciones básicas de toma de mediciones, visualización de estas y comunicación con una plataforma del Internet de las Cosas por vía inalámbrica, se diseña el sistema operativo que permite el funcionamiento deseado del dispositivo según los objetivos establecidos al inicio del proyecto.

Finalmente, se perfilan los detalles en el diseño final del medidor para obtener un dispositivo utilizable por cualquier usuario, y se valoran las limitaciones y posibles mejoras del producto conseguido y la posibilidad de este de competir con productos ya existentes.

Tabla de contenido

1. Introducción

El control de la calidad del aire en espacios públicos, especialmente en entornos cerrados, ha ganado una gran importancia recientemente debido a la aparición del Covid-19. Sus consecuencias han afectado a todos los ámbitos de la vida cotidiana y profesional de la población.

Uno de los sectores más afectados por la pandemia ha sido el educativo. En las aulas se acumula un gran número de personas durante largos periodos de tiempo, lo que ha obligado a tomar una serie de medidas para intentar prevenir la transmisión del virus.

Estas medidas vienen dadas directamente por el Gobierno de España, concretamente el Ministerio de Sanidad y el Ministerio de Educación y Formación Profesional. Se pueden clasificar las medidas en cuatro principios básicos. [1]

- Limitación de contactos: mantener distancias mínimas entre individuos y controlar el número de personas que se concentran en un mismo espacio.
- Medidas de prevención personal: fomentar principalmente el uso de mascarillas y la correcta higiene de manos.
- Limpieza y ventilación: se intensificará la desinfección de espacios y material académicos, y se controlará minuciosamente la ventilación de estos.
- Gestión de los casos: aislamiento y cuarentena de individuos que hayan contraído el virus o hayan estado en contacto directo con alguien que lo haya hecho.

Como se puede ver, la correcta ventilación de los espacios académicos es una herramienta fundamental para combatir la transmisión del virus, debido a la evidencia sobre la transmisión del Covid-19 por aerosoles. Por lo tanto, es necesario controlar en cada momento la efectividad de esta ventilación para saber si un espacio es seguro para su uso por parte del público.

En cuanto a cómo se puede medir la ventilación de un aula, la solución más sencilla es medir el dióxido de carbono en el aire, ya que este gas es el producto de la respiración humana. De esta forma, una baja concentración de dióxido de carbono es una indicación directa de una buena ventilación.

La tecnología de medición del dióxido de carbono está muy desarrollada, con una gran multitud de dispositivos de diferentes calidades y precios en el mercado. Sin embargo, ninguno de ellos se ha diseñado con el objetivo específico de ser usado en el ámbito académico, por lo que cabe preguntarse si sería interesante diseñar un producto que cumpla las necesidades específicas para este uso por un precio de fabricación lo más asequible posible. Esta es la idea principal en la que se basa este proyecto.

1.1. Estudio de mercado

Antes de empezar con el diseño del dispositivo, es necesario establecer cuáles son las funciones que este deberá ser capaz de llevar a cabo. Estas especificaciones vendrán dadas por las necesidades que se quieran satisfacer con su uso, pero también es beneficioso estudiar las capacidades de otros dispositivos similares ya existentes.

A continuación, aparecen algunos de los modelos de medidor de CO2 más populares y las características que ofrecen.

- Medidor de CO2 PCE-AQD 50: [2]
  - Medición de concentración de CO2, temperatura, humedad y presión.
  - Sistema de alarma visual y acústica.
  - Rango de medida: 0-40.000 PPM.
  - Historial de medidas por pantalla.
  - Registro de datos en tarjeta microSD.
  - Precio: 479 €.

- Medidor de CO2 Dioxcare Winix: [3]
  - Medición de concentración de CO2, temperatura y humedad.
  - Alarma en caso de exceso.
  - Rango de medida: 0-9.999 PPM.
  - 9 horas de autonomía.
  - Posibilidad de exportar datos al ordenador como PDF.
  - Calibración automática con opción de calibración manual.
  - Dimensiones: 140 x 134 x 33 mm.
  - Precio: 119 €.

- Medidor CO2 Digital VDL: [4]
  - Concentración de CO2, temperatura, humedad, fecha y hora en pantalla.
  - Alarma visual en caso de exceso.
  - Rango de medida de CO2: 0-9.999 PPM.
  - Memoria de lectura máxima y mínima.
  - Dimensiones: 120 x 134 x 54 mm.
  - Precio: 199 €.

- Medidor CO2 Básico: [5]
  - Medición de concentración de CO2 y temperatura.
  - Rango de medida de CO2: 0-3.000 PPM.
  - Dimensiones: 116 x 24 x 42 mm.
  - Precio: 99 €.

1.2. Objetivos

A partir del estudio de mercado realizado, se pueden establecer una serie de requisitos mínimos que un nuevo dispositivo debería cumplir para competir con los modelos ya existentes. Estos requisitos vienen expresados a continuación.

- Datos por pantalla de concentración de CO2 y temperatura en tiempo real.
- Sistema de alarma en caso de exceso.
- Autonomía suficiente para su uso portátil.
- Registro del historial de medidas.
- Posibilidad de exportar y tratar datos por vía inalámbrica.
- Dimensiones aptas para su cómodo uso y transporte (idealmente 15 cm como dimensión máxima).

El objetivo principal de este proyecto es diseñar y desarrollar un dispositivo que cumpla estos requisitos mínimos, con la posibilidad de añadir otros durante su desarrollo, por un precio más asequible que el de los dispositivos ya existentes, centrando su diseño enteramente en un uso en espacios académicos. Simultáneamente, se desarrollará un prototipo para obtener al finalizar el proyecto un producto que pueda ponerse a prueba en casos prácticos.

Además de este objetivo principal, se pretenden alcanzar otros objetivos de carácter académico:

- Afianzar conocimientos adquiridos anteriormente como parte del grado realizado, en concreto aquellos relacionados con electrónica y programación de software.
- Adquirir destreza y nuevos conocimientos sobre programación en el entorno de Arduino.
- Afianzar la capacidad de diseñar un producto en base a unas especificaciones u objetivos iniciales.
- Desarrollar la capacidad de observar de forma crítica un proyecto propio y compararlo con otros existentes.
- Adquirir destreza para la búsqueda de información sobre diferentes temas en fuentes diversas.

1.3. Metodología

Establecidas las especificaciones de diseño del dispositivo, el primer paso será elegir sus componentes, escogiendo la mejor opción tras comparar diferentes posibilidades.

Una vez elegidos y obtenidos los componentes, comenzará el proceso de montaje y programación de un prototipo, explicando cada paso hasta obtener un primer dispositivo que sea capaz de llevar a cabo las funciones principales de nuestro objetivo. El lenguaje de programación a utilizar será C++ en la plataforma de Arduino.

Tras el montaje de este prototipo, se procederá a perfeccionar su diseño y añadir lo necesario para obtener un producto finalizado y utilizable por el usuario.

Por último, se revisará el resultado obtenido, tanto sus capacidades y ventajas como sus limitaciones, y se obtendrán unas conclusiones y, en su caso, posibilidades de mejora.

2. Selección de componentes

Los componentes principales que formarán el dispositivo serán un microcontrolador, un sensor de concentración de CO2 y una pantalla para la visualización de datos. En este apartado se estudiarán las diferentes opciones para cada componente y se justificará la elección de cada uno según sus características.

2.1. Microcontrolador

Como se ha indicado anteriormente, la plataforma en la que se va a desarrollar este proyecto será Arduino. La elección de este software se debe principalmente a su sencillez de uso, el ser una plataforma gratuita y su compatibilidad con la mayoría de sistemas operativos y componentes electrónicos en el mercado. Además, al ser Arduino una plataforma de código abierto, se puede encontrar una gran variedad de información, foros y respuestas sobre posibles problemas que pudieran surgir a lo largo del desarrollo del proyecto.

Un modelo de placa de Arduino básico, como podría ser el Arduino Uno, sería capaz de realizar la mayoría de las funciones que se requieren para el funcionamiento de este dispositivo. Sin embargo, para que el medidor sea capaz de enviar información de forma inalámbrica sería necesario añadir un módulo externo que permitiera esta conexión, ya sea Bluetooth o Wifi. Una opción más adecuada para estas necesidades es utilizar directamente una placa **NodeMCU**.

El término NodeMCU recoge tanto un firmware de código abierto como una serie de placas de desarrollo diseñadas específicamente para su uso en el Internet de las Cosas. Estas placas tienen incorporado un **ESP8266**, un chip Wifi de bajo costo, el cual permite que el microcontrolador se conecte vía Wifi sin necesidad de añadirle módulos externos.

Dentro de las placas NodeMCU, existen varias versiones que se han ido desarrollando por distintos fabricantes. A parte de la ya obsoleta primera versión 0.9, existen las versiones 1.0/V2 (segunda generación) y 1.0/V3 (tercera generación). Esta última es una versión mejorada de la anterior, pero las mejoras que ofrece no son particularmente útiles para este proyecto, por lo que se utilizará una placa NodeMCU de segunda generación. En concreto, se escoge una placa del fabricante Amica, cuyas características se especifican a continuación. [6]

- Programabilidad con el IDE de Arduino.
- Alimentación vía USB.
- Voltajes de salida de 3,3 y 5V.
- 13 pines GPIO.
- LED y botón de Reset integrados.
- Protocolos de comunicación en serie: UART, I2C y SPI.
- Precio: 12 €.

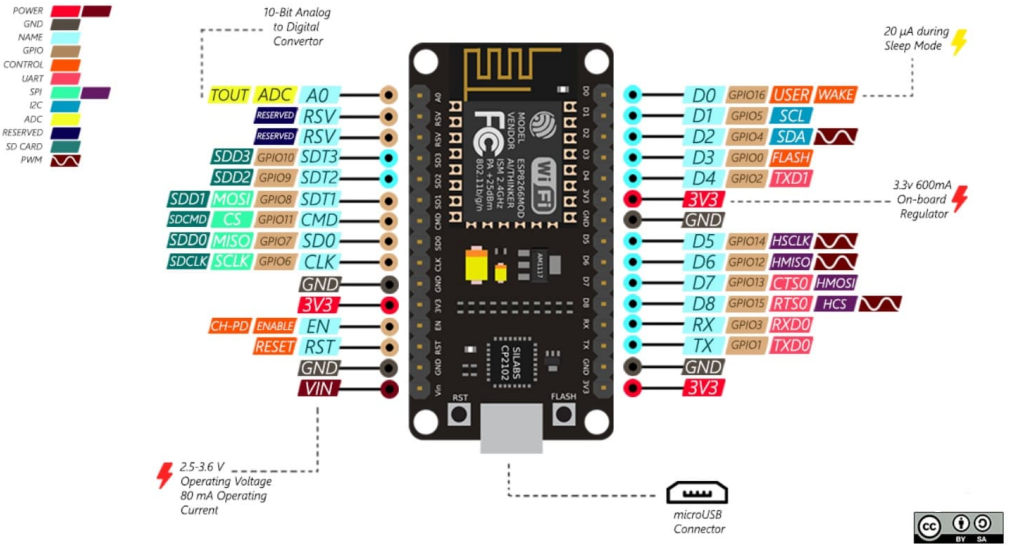

Ilustración 1: Estructura del NodeMCU

Este será el microcontrolador al que se conecten el resto de los componentes del dispositivo, y en el que se programará el código a través de el IDE de Arduino. Así, los demás componentes se elegirán con la condición de que sean compatibles con esta placa.

2.2. Sensor de concentración de CO2

Existen en el mercado un gran número de sensores de calidad del aire, diferenciándose en la información que son capaces de medir y el sistema que utilizan para ello. En este apartado se estudiarán las diferentes opciones para el medidor y se decidirá razonadamente por la que mejor se adapte a sus necesidades.

En primer lugar, hay que analizar las diferentes formas de las que se puede medir la concentración de un gas. Según el método que utilizan para detectar partículas en el aire, los sensores pueden ser electroquímicos, piezoeléctricos u ópticos. Debido a su simplicidad y su precisión, la gran mayoría de sensores son de este último tipo.

Los sensores ópticos emiten luz infrarroja de una longitud de onda determinada, de modo que esta es absorbida por las partículas de CO2. Tras atravesar un pequeño espacio ocupado por el aire a medir, la intensidad de esta luz es medida en un receptor, de forma que, a mayor intensidad, menos radiación ha sido absorbida, lo que indica una menor concentración de la partícula a medir en el aire. Finalmente, el sensor convierte este valor de radiación en una señal eléctrica equivalente a la concentración del gas en partes por millón (PPM). [7]

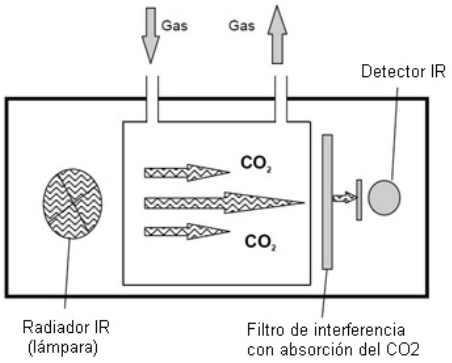

Ilustración 2: Sistema de medición por infrarrojos [8]

A continuación, hay que decidir qué información necesitamos que nos proporcione el sensor. A parte de medir la concentración de CO2 en el ambiente, existen sensores que miden la concentración de otras sustancias, como aerosoles, humedad u otras partículas, proporcionando así una lectura completa de la calidad del aire. Para cumplir los requisitos de este proyecto, el sensor solo necesita medir la concentración de CO2, ya que a partir de esta se puede controlar el nivel de ventilación sin necesidad de más datos.

Además de la concentración de partículas, los sensores pueden medir datos de temperatura y presión, ya que la medición por infrarrojos se ve afectada por estas propiedades del aire conforme a la ley de los gases ideales. En concreto, un aumento de temperatura hace que el sensor detecte una menor cantidad de moléculas en el aire, mientras que si la presión aumenta el sensor detectará una mayor concentración. La compensación de los resultados con estos datos es fundamental para que la medición sea precisa, por lo que muchos sensores son capaces de medir como mínimo la temperatura, y algunos compensan automáticamente el efecto de esta en la medición del CO2.

Un requisito indispensable es que el sensor sea compatible con el microcontrolador elegido. La placa NodeMCU es capaz de recibir una gran variedad de señales, entre ellas los tres protocolos principales de comunicación en serie (UART, I2C y SPI), por lo que esto no debería ser un problema. Por último, la decisión entre un sensor u otro dependerá de la relación entre su calidad (rango y precisión de medida, potencia consumida, dimensiones y peso) y su precio.

En base a todo lo anterior, el sensor escogido para este proyecto es el modelo **MH-Z19b** de la empresa **Winsen**. A continuación, se enumeran sus principales características.[1] [9]

- Detección de partículas de CO2 por infrarrojos.
- Toma de temperatura y compensación automática.
- Dimensiones: 33mm x 20 mm x 9 mm.
- Peso: 5 gramos.
- Señales de salida analógica, PWM y UART.
- Calibración manual y automática.
- Voltaje de alimentación: 4,5 – 5,5 V.
- Intensidad de alimentación media: 60 mA.
- Rango de medida: 0 – 5.000 ppm con posibilidad de cambiarlo.
- Rango de temperatura: 0 – 50 ºC.
- Precisión: ± (50 ppm + 3% del valor medido).
- Precio: 17 €.

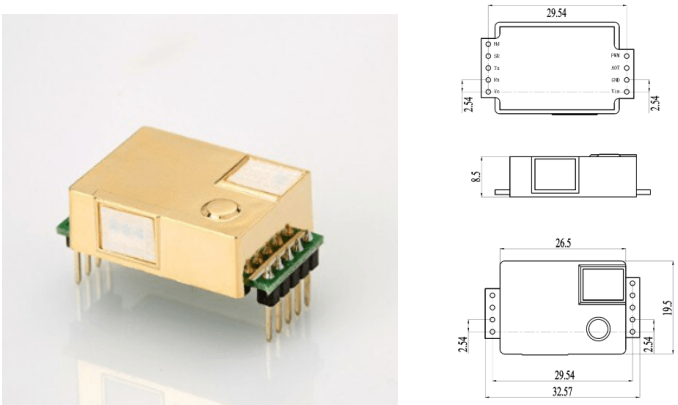

Ilustración 3: Sensor MH-Z19b

[1] Otros modelos de sensor considerados y sus características pueden encontrarse en el anexo 2.

Otros sensores tenían un precio algo inferior, pero la menor seguridad y precisión de los resultados y la escasez en algunos casos de datos técnicos del sensor hacen que esta sea la opción más adecuada. En cuanto al rango de medida, este es más bajo que el de alguno de los dispositivos que aparecieron en apartados anteriores. Sin embargo, los niveles de CO2 habituales no suelen alcanzar las 5.000 PPM, por lo que este rango es suficiente. En apartados siguientes se explicará más en detalle este sensor y sus resultados a la hora de conectarlo al microcontrolador.

2.3. Pantalla

El último de los componentes principales es una pantalla a través de la cual se puedan visualizar los datos que mide el sensor. En este caso, lo principal a la hora de elegir la pantalla serán las dimensiones de ésta, el número de píxeles y el protocolo de comunicación con el microcontrolador.

Principalmente, podemos usar dos tipos de pantalla, OLED o LCD. La ventaja principal de las pantallas OLED con respecto a las otras es que permiten una mayor libertad a la hora de mostrar imágenes por pantalla, ya que el control se realiza píxel a píxel, a diferencia de las LCD en las que solo se puede variar el carácter que aparece en cada espacio. Además, la conexión con una pantalla OLED es muy simple, necesitando solo cuatro pines para recibir la información desde el microcontrolador.

En cuanto al tamaño de la pantalla, debemos tener en cuenta el texto que se pretende mostrar en ella. Este texto no será demasiado largo, necesitando solo espacio para dos o tres palabras ("Concentración: 400 PPM", por ejemplo), por lo que un número estándar de 128 píxeles de longitud será suficiente. Ahora bien, en caso de que se quiera escribir varias líneas de texto a la vez, lo cual será de gran utilidad, serían necesarios 64 píxeles de altura, en lugar de los 32 píxeles de las pantallas más pequeñas. Así, se elige una pantalla de 1,3 pulgadas con 128x64 píxeles.

El modelo elegido finalmente es el **AZ-Delivery OLED I2C de 1,3 pulgadas,** cuyas especificaciones aparecen a continuación.[2]

- Voltaje de alimentación: 3,3 / 5 V.
- Intensidad de alimentación: < 11 mA.
- Interfaz de comunicación I2C.
- Tamaño de pantalla:
- Dimensiones: 36mm x 34mm x 3 mm.
- Resolución: 128 x 64 píxeles.
- Precio: 8 €.

[2] Otros modelos de pantalla considerados y sus características pueden encontrarse en el anexo 3.

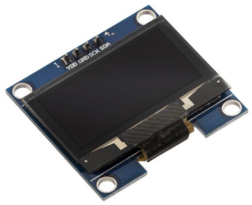

Ilustración 4: Pantalla OLED I2C 1,3 pulgadas

Al igual que con el sensor, el funcionamiento y la estructura de la pantalla se explicarán más a fondo en apartados siguientes, cuando se lleve a cabo su conexión y se programe su código correspondiente.

3. Diseño del prototipo

En los siguientes apartados se describirá el proceso de programación del código que haga funcionar al dispositivo una vez se dispone de los componentes que lo forman. Así, se explicará paso a paso la evolución del código y el montaje del dispositivo hasta conseguir un sistema capaz de medir correctamente la concentración de CO2 en el ambiente y enviar los datos a una plataforma a través de una conexión inalámbrica.

3.1. Sistema de medición

El primer paso en la programación del dispositivo consistirá en hacer funcionar al sensor, de forma que mida las variables que se quieren obtener y envíe a la placa NodeMCU los datos resultantes de la medición. Cuando el microcontrolador reciba esta información, esta se visualizará en la pantalla del ordenador utilizado para cargar el código.

El sensor debe conectarse a un voltaje de entre 4,5 y 5,5 Voltios para funcionar, por lo que se conectará el pin Vin del sensor (pin de alimentación) con el pin Vin del microcontrolador (salida de aproximadamente 5 V), y el pin GND del sensor con uno de los pines de masa de la placa. Así, el sensor ya estaría alimentado, por lo que falta establecer la conexión de datos entre los dispositivos.

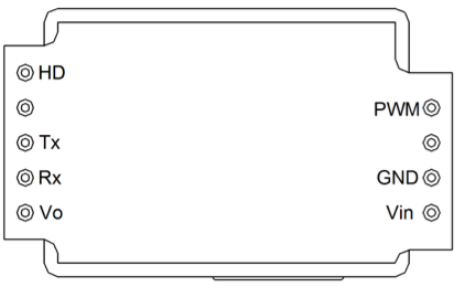

Ilustración 5: Pines del sensor MH-Z19b

Este sensor cuenta con tres métodos diferentes de enviar información: por señal analógica, por señal PWM y por comunicación UART. A continuación, se explica cómo recibir e interpretar cada tipo de señal.

- **Señal analógica:** a través del pin V0, el sensor alcanza un voltaje equivalente a la concentración de CO2 medida. Este voltaje se moverá entre los 0 V, cuando se lea el valor mínimo de concentración (400 PPM), y los 3,3 V, cuando se alcance el rango máximo (5000 PPM por defecto). El microcontrolador lee este voltaje y lo interpreta mediante un conversor analógico como una variable entre 0 y 1023, lo que resulta en una resolución de 1024. Una vez se obtiene este valor, es posible establecer una relación con el rango de medidas para obtener una aproximación de la concentración de CO2. El problema de esta opción es que 1024 es una resolución muy baja cuando el rango de medida es tan amplio, por lo que los resultados no serían suficientemente precisos.

- **Señal PWM:** utilizando en pin PWM del sensor, obtenemos una señal periódica cuyo ciclo de trabajo varía en función de la concentración medida. Esta señal se mueve entre dos valores de voltaje, 0 V y 3,3 V. Lo que cambia con la concentración es el tiempo que permanece con voltaje alto, lo que se suele llamar el ancho de pulso. Así, midiendo el tiempo que la señal está en alto y la duración total del ciclo, se obtendría el ciclo de trabajo, es decir, la relación entre estos valores, que irá desde el 0 cuando la concentración sea mínima hasta el 1 cuando esta sea máxima. De igual forma que en el caso anterior, es posible establecer una relación para obtener un valor, en este caso más preciso, de la concentración.

- **Señal UART:** el último de los métodos se basa en la conexión UART, un protocolo de comunicación en serie que permite recibir y enviar información a través de dos pines, el **Tx (transmisor)**, que deberá conectarse al receptor de otro dispositivo, y el **Rx (receptor)**, que deberá conectarse a un transmisor. De esta forma, el microcontrolador recibe directamente los valores de concentración de CO2 y temperatura sin tener que modificarlos con anterioridad. Además, las señales son más fiables al evitarse el ruido y las caídas de tensión en las mediciones analógicas.

El método que se utilizará para este dispositivo, debido a sus ventajas respecto a los otros, es la comunicación UART. Para ello, es necesario establecer un nuevo serial que comunique los pines del sensor con una pareja Tx/Rx del microcontrolador. Los pines Tx y Rx principales de la placa se reservan para la comunicación con la IDE de Arduino, por lo que no son una opción válida. En su lugar, se emplearán los **pines D7 y D8** como Rx y Tx, respectivamente, y se creará un nuevo serial que funcione a una velocidad de 9600 Baudios. El pin Tx del microcontrolador se conectará al Rx del sensor, y viceversa.

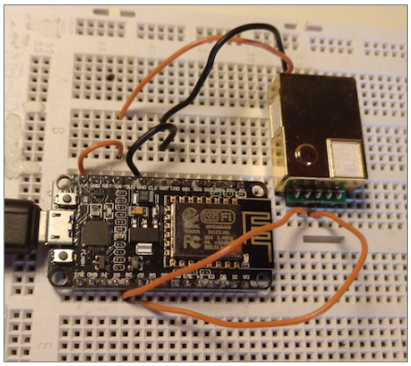

Ilustración 6: Conexión del NodeMCU con el sensor

Una vez establecida la conexión, es posible enviar comandos al sensor y recibir de este la información necesaria. Estos comandos consisten en secuencias de números en base hexadecimal especificadas en la hoja de datos del sensor. Sin embargo, existen varias librerías de Arduino creadas para facilitar el uso de sensores del tipo MH-Z, de forma que los comandos se simplifican al no ser necesario escribir directamente las secuencias anteriores. Una de ellas, la **librería MH-Z19**, desarrollada por Jonathan Dempsey, será la que se utilice para comunicar órdenes al sensor de aquí en adelante. [10]

Esta biblioteca contiene comandos simples que obtienen directamente las lecturas de concentración de CO2 ("getCO2()") en PPM y temperatura ("getTemperature()") en grados Celsius, y las expresan como números enteros, por lo que solo es necesario crear un par de variables del tipo adecuado y otorgarles estos valores.

Para que la medición sea en tiempo real, se deben actualizar continuamente las mediciones, por lo que se programará el código de forma que este repita estas operaciones cada vez que pase un tiempo establecido. El sensor toma una nueva medida cada cinco segundos, así que esta será la máxima frecuencia a la que se pueden actualizar los resultados. Tomando como referencia temporal el instante en el que se toma una medición, se crea un contador, de forma que se vuelva a tomar medidas cuando el tiempo transcurrido alcance los cinco segundos.

Por último, se visualizan estos datos con el monitor serie del IDE de Arduino en la pantalla del ordenador, de forma que se obtiene un nuevo valor de concentración y temperatura cada cinco segundos.

```
Concentración CO2: 450
Temperatura: 30
Concentración CO2: 452
Temperatura: 30
Concentración CO2: 452
Temperatura: 30
Concentración CO2: 454
Temperatura: 30
Concentración CO2: 455
Temperatura: 30
Concentración CO2: 455
Temperatura: 30
Concentración CO2: 457
Temperatura: 30
Concentración CO2: 459
```

Ilustración 7: Valores de concentración y temperatura

Ya se ha obtenido un sistema que obtiene los valores buscados de concentración de CO2 y temperatura con una frecuencia determinada. El código que hace funcionar este primer sistema se puede encontrar en uno de los anexos. En apartados siguientes se desarrollarán las diferentes formas de visualización de resultados y se analizará la precisión de estos datos.

3.2. Visualización de datos por pantalla

El siguiente paso consiste en visualizar los datos que llegan desde el sensor en una pantalla OLED, en lugar de usar el serial del ordenador. La pantalla necesita un voltaje de 3,3 V para funcionar, pero tiene incorporado un regulador de tensión que permite conectarla al mismo pin de 5 V que alimenta al sensor. Así, al conectar el pin VDD de la pantalla con el pin Vin de la placa, y el pin GND con la masa del microcontrolador, la pantalla ya está alimentada.

La pantalla recibirá datos desde el microcontrolador por medio de la interfaz **I2C**. Este protocolo de comunicación en serie permite coordinar y enviar información desde un dispositivo maestro a un periférico, utilizando solamente dos pines: un SCK, empleado como señal de reloj, y un SDA, utilizado para el intercambio de datos. Estos pines deberán ir conectados a los pines **SDA (D1)** y **SCL (D2)** del microcontrolador, respectivamente.

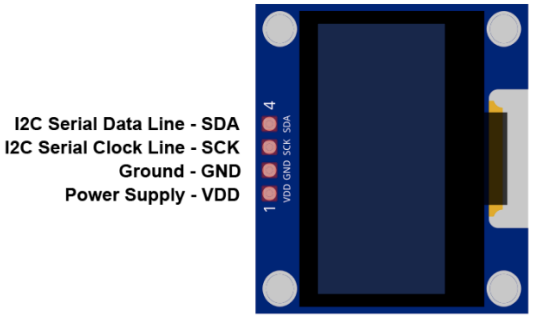

Ilustración 8: Pines de la pantalla OLED

Una vez establecidas todas las conexiones, solo falta programar el funcionamiento del sistema. Al igual que pasaba con el sensor, existen librerías que simplifican esta programación, al ofrecer comandos simplificados para el uso general de pantallas similares. Para nuestro sistema, utilizaremos la **librería U8g2** recomendada por el fabricante de la pantalla en su manual de uso. [3] [11]

Antes de nada, es necesario crear un objeto que se utilizará para controlar la pantalla. Según el modelo de esta, se deberá que escribir una línea de código diferente, en la cual también se señalan los pines de la placa que cumplirán las funciones de SDA y SCL. En este caso, esta línea será la siguiente:

[3] Un ejemplo de código utilizado como referencia para este proyecto aparece en el manual de usuario de AZ-Delivery que se recibe al comprar la pantalla.

```
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE,D1,D2);
```

Antes de enviar cualquier información a la pantalla, es necesario borrar la información que se ha enviado anteriormente, lo que se denomina el buffer de datos, lo cual se hace usando el comando "clearBuffer()". A continuación, se debe especificar el contenido que se quiere visualizar en la pantalla, y enviar este nuevo buffer de datos por medio del comando "sendBuffer()". Si se quisiera visualizar a continuación otro contenido diferente, es necesario volver a limpiar el buffer antes de enviar nueva información.

En las pantallas OLED se pueden visualizar una infinidad de diseños y figuras diferentes, pero para este dispositivo solo es necesario visualizar líneas de texto compuestas por letras y números. El comando utilizado para ello es el llamado "drawStr()", en el cual se debe especificar la posición del inicio del texto en coordenadas x e y, y el texto que queremos escribir en pantalla. Este texto se expresa como una variable de tipo String, por lo que, para visualizar las mediciones del sensor, debemos crear unas nuevas variables de este tipo y darles los valores correspondientes.

Por último, utilizaremos una función, denominada "u8g2_prepare()", que establece una serie de preferencias previas, en concreto el tipo de letra, el color del texto y el fondo (blanco y negro), la posición y la dirección en la que se escribirán las líneas de texto.

Teniendo todo esto en cuenta, se modifica el código desarrollado en el apartado anterior, que mostraba los resultados por el serial del ordenador, para que los muestre ahora por la pantalla OLED conectada al microcontrolador. De nuevo, se muestra un nuevo valor cada cinco segundos, tras cada nueva medición del sensor.

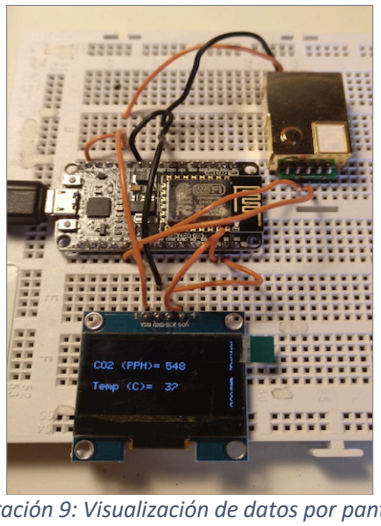

Ilustración 9: Visualización de datos por pantalla

3.3. Comunicación inalámbrica

Ya se dispone de un sistema que mide la concentración de CO2 y la temperatura en tiempo real y muestra estos resultados por una pantalla. El siguiente paso es establecer una conexión inalámbrica con una plataforma online en la que se puedan visualizar estos datos sin conexión física al microcontrolador.

Existen multitud de plataformas de Internet de las Cosas en el mercado, capaces de recibir información de un dispositivo vía Wifi y tratar los datos transmitidos con una infinidad de herramientas diferentes. A la hora de elegir una de ellas para este proyecto, es necesario especificar qué se requerirá de esta plataforma.

Se necesita una plataforma en la que poder visualizar las mediciones de sensor en tiempo real, accesible desde cualquier dispositivo con conexión a Internet. Esta no es una necesidad muy compleja, por lo que la mayoría de plataformas existentes servirían. Otro factor fundamental es el precio. Al basarse este proyecto en diseñar un dispositivo lo más económico posible, interesaría elegir una plataforma gratuita. Tras estudiar diferentes opciones, nos decidimos por usar la versión **Demo de Thingsboard**. [12]

Thingsboard es una plataforma IOT especializada en la visualización de datos, contando con una gran cantidad de gráficas y widgets personalizables que serán de gran utilidad. Aunque la versión completa es de pago, existe una versión Demo gratuita que ofrece herramientas suficientes para el alcance de este proyecto, la cual será la que se utilice de aquí en adelante.

Tras crear una cuenta en Thingsboard, es posible empezar a utilizar esta plataforma. El uso de Thingsboard es bastante sencillo, y existen librerías que simplifican la programación del software. Para establecer la conexión del microcontrolador con la plataforma, utilizaremos la **librería ThingsBoard**, creada por la propia empresa. De nuevo, se modifica el código desarrollado hasta ahora en apartados anteriores para añadir lo necesario para el envío inalámbrico de información. [13]

En primer lugar, es necesario crear un nuevo dispositivo en la web de Thingsboard, de tipo predeterminado, y hacerlo público para que se puedan visualizar los resultados al final. Este dispositivo recién creado tiene una credencial, denominada token de acceso, que lo identifica. En el nuevo código, se deberá que especificar este código, además de los datos de la red Wifi a utilizar, de la siguiente forma.

```
1. //Datos de la red Wifi
2. #define WIFI_AP                 "Aquaris X"
3. #define WIFI_PASSWORD           "123456ABC"
4.
5. //Datos del dispositivo en Thingsboard
6. #define TOKEN                  "v42uu5eMqaqljx6gwyRC"
7. #define THINGSBOARD_SERVER      "demo.thingsboard.io"
```

La conexión utilizada en un primer momento vendrá de un dispositivo móvil, pero bastará con cambiar estos datos para conectarnos a cualquier red. En el dispositivo final, se deberían establecer los datos de la red Wifi del centro académico en el que se vaya a utilizar el medidor.

A continuación, se inicia la conexión con la red Wifi y con Thingsboard a partir de los datos anteriores. Además, se añadirá una función en el bucle, a la que llamaremos "reconnect()", para que el sistema vuelva a conectarse a la red Wifi en caso de que se pierda la conexión en algún momento.

El comando que permite enviar información a Thingsboard es "sendTelemetryString()", en el que se debe especificar un nombre para la información a enviar y el valor de esta. En este caso, se enviarán dos telemetrías de datos diferentes, una con las mediciones de concentración de CO2 y otra con las de temperatura. Añadiendo estos comandos al bucle de lectura de datos, se consigue que llegue nueva información al dispositivo en Thingsboard cada cinco segundos.

Una vez que la información llega a Thingsboard, se puede crear un nuevo panel, al que se denominará "Medidor de CO2", que de nuevo deberá ser público. Este será el panel que se podrá abrir desde cualquier dispositivo, y en el que se podrá visualizar los datos que se prefieran de forma personalizada. Para añadir contenido a este panel, basta con añadir alguno de los widgets que ofrece Thingsboard. También se podría crear un nuevo widget, pero para expresar la información necesaria para este proyecto bastará con los ya existentes en la plataforma.

Para visualizar las mediciones de nuestro dispositivo, se añadirán al panel dos marcadores con los últimos valores de concentración y temperatura, y una gráfica temporal en la que ver la evolución de la concentración a lo largo del tiempo. Estas gráficas son totalmente personalizables, por lo que es posible ajustarlas para conseguir la máxima claridad en los resultados. A continuación se puede ver el contenido de este panel.

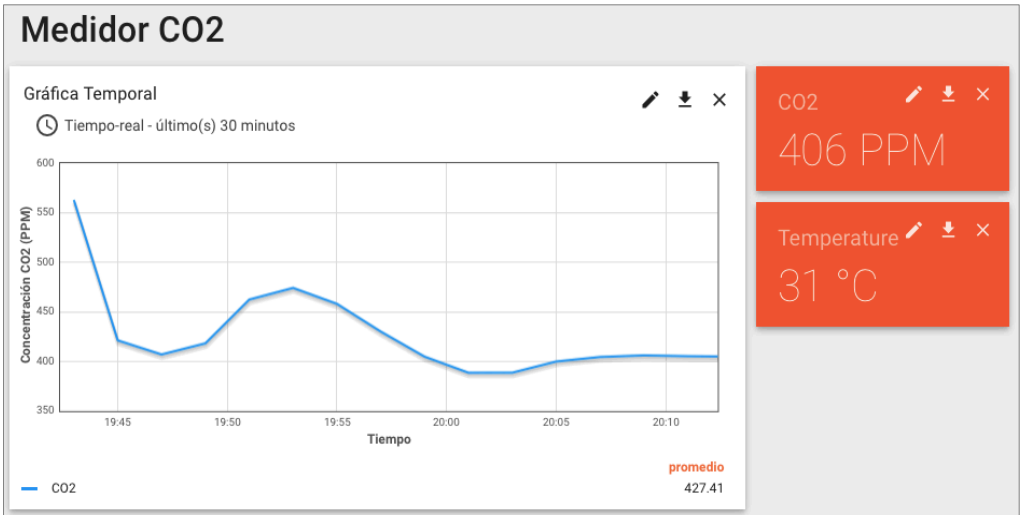

Ilustración 10: Panel Thingsboard del dispositivo

Para acceder a este panel desde cualquier dispositivo, ya sea móvil o PC, basta con entrar a la dirección URL que se creó con el mismo:

https://demo.thingsboard.io/dashboard/64db20d0-adcf-11eb-9729-914d29c4aef9?publicId=c7ea2160-acff-11eb-83ac-bf5e0e990b9e

Llegados a este punto, se ha conseguido crear un dispositivo que mide la concentración de CO2 en tiempo real y envía los resultados a una plataforma virtual accesible desde cualquier dispositivo con conexión Wifi. El último paso en el diseño de este primer dispositivo es analizar la exactitud de estas mediciones, lo cual se realizará a continuación.

3.4. Calibración y comprobación de resultados

El último paso para la obtención del prototipo consiste en analizar los resultados que este proporciona, y ajustarlos en caso de que sea necesario para alcanzar una mayor precisión.

Un aspecto fundamental a la hora de obtener resultados precisos es la **calibración del dispositivo**, es decir, la referencia que utiliza el sensor para dar valores a la concentración medida. El sensor utilizado emplea una referencia de punto cero, es decir, toma una medida, a la que le otorga un valor mínimo de concentración, y realiza el resto de las medidas comparándolas con ella. Si esta referencia no es exacta, ningún resultado de las mediciones lo será.

El valor de punto cero del sensor es 400 PPM, aproximadamente la concentración de CO2 en el exterior en condiciones normales, por lo que para llevar a cabo la calibración se debe tomar una medida en estas condiciones para usarla como referencia. Según la hoja de datos del sensor, existen dos formas de tomar esta medida.

- **Calibración automática**: si no se especifica lo contrario, el sensor se calibra de forma automática. Esto significa que, cada 24 horas de funcionamiento sin pausa, se toma el mínimo valor que se haya registrado en ese periodo, registrándolo como referencia de 400 PPM. Evidentemente, para que este método funcione correctamente el sensor debe estar expuesto a condiciones similares a las del exterior por lo menos una vez al día, ya que si no el valor mínimo medido será superior a los 400 PPM, falsificando los resultados posteriores.

- **Calibración manual**: utilizando el comando “calibrate()” el sensor toma una medida instantánea y la registra como referencia de 400 PPM. Así, basta con exponer al sensor a una concentración de aproximadamente 400 PPM, es decir, condiciones normales en el exterior, durante un tiempo determinado para realizar la calibración. Según la hoja de datos, el tiempo que el sensor debería permanecer en el exterior para alcanzar con seguridad una medida de 400 PPM es 20 minutos.

En apartados anteriores, al no especificar el método de calibración, esta era automática por defecto. La utilización de uno u otro método de calibración dependerá del uso que se le quiera dar al dispositivo. Si se quiere medir la concentración de CO2 en un espacio fijo, con ventilación frecuente, durante largos periodos de tiempo, como podrían ser varios días o semanas seguidas, la calibración automática es más cómoda. Sin embargo, si se pretende utilizar el dispositivo durante un tiempo más reducido, o se quiere medir la concentración en un espacio en el que la ventilación nunca es completa, entonces se deberá realizar una calibración manual previa en condiciones similares al exterior.

Otro parámetro importante en la toma de medidas es el **rango de medidas** que abarca el sensor. Por defecto, este rango va desde las 400 PPM hasta las 5.000 PPM. Sin embargo, este parámetro puede modificarse, usando el comando "setRange()". La utilidad de cambiar el rango de medidas se basa en que cuanto mayor es el rango, menor es la resolución de los resultados, por lo que si ajustamos el rango a los valores que realísticamente se van a encontrar, es posible aumentar la precisión de las medidas.

Los niveles de concentración de CO2 rondan las 400 PPM en espacios abiertos y los 600 PPM en espacios cerrados. Normalmente, una concentración de 1.000 PPM se considera el límite tolerable en espacios cerrados para una estancia prolongada, no siendo hasta las 5.000 PPM cuando se puedan empezar a dar señales de malestar físico. A parte, la aparición del Covid-19 da una mayor importancia a la ventilación de espacios cerrados, por lo que el valor máximo recomendable para un espacio público cerrado estará por debajo de las 1.000 PPM.

En cualquier caso, nunca se deberían alcanzar valores muy superiores a las 1.000 PPM en espacios académicos, por lo que el rango de 5.000 PPM parece excesivamente elevado. Bajando este valor a **2.000 PPM** se obtienen resultados más precisos a cambio de no poder medir valores tan altos que nunca se llegarán a dar en el uso normal del dispositivo.

Finalmente, para comprobar el correcto funcionamiento del dispositivo, conviene tomar medidas durante un tiempo prolongado y estudiar la evolución de los resultados. De esta forma, es posible comprobar que los datos obtenidos corresponden a las condiciones en las que trabaja el medidor en cada momento.

A continuación, aparece la evolución de la concentración de CO2 en una habitación con una persona en su interior a lo largo de 12 horas tal y como aparece en el panel de Thingsboard.

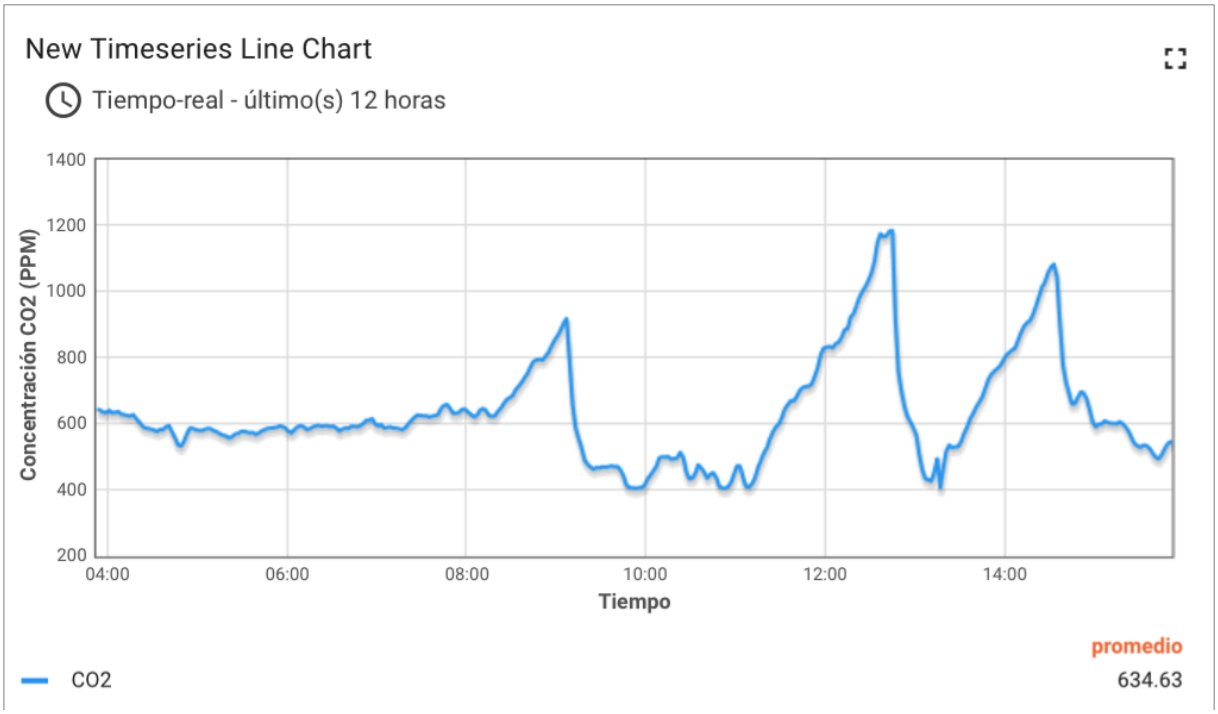

Ilustración 11: Resultados de la medición en 12 horas

Al inicio de este periodo, el medidor se encuentra en la habitación con una puerta abierta, comunicándola con otra habitación de más tamaño, y se mantienen valores de concentración constantes alrededor de las 600 PPM. Sobre las 8:00 horas, se cierra la puerta, de forma que el nivel de CO2 aumenta gradualmente debido a la respiración de la persona en la habitación. A las 9:00 horas, cuando la concentración ha alcanzado las 900 PPM, se abre una ventana al exterior, de forma que la habitación se ventila rápidamente hasta llegar a las 400-500 PPM propias del exterior en condiciones normales.

Alrededor de las 11:00 se vuelve a cerrar la ventana, de forma que la concentración de CO2 comienza a elevarse. Al permanecer esta vez más tiempo la habitación cerrada, se alcanzan las 1200 PPM antes de volver a abrir la ventana cerca de las 13:00 horas, tras lo cual se vuelve a valores cercanos a las 400 PPM.

Finalmente, se vuelve a cerrar la ventana, dejando que los niveles suban hasta casi 1.100 PPM, para después abrir la puerta. Una vez abierta, los niveles de CO2 bajan hasta estabilizarse de nuevo alrededor de las 600 PPM.

Viendo esta evolución y comparándola con las condiciones a las que se ha sometido el sensor, se ve que este detecta de forma muy precisa los cambios graduales en la concentración de CO2 a lo largo del tiempo.

Para comprobar la respuesta del sensor a cambios más rápidos del nivel de CO2, se realizan varias pruebas consistentes en respirar repetidas veces directamente sobre el dispositivo para ver si registra estos aumentos súbitos de concentración. El resultado de una de estas pruebas se puede ver en la gráfica siguiente.

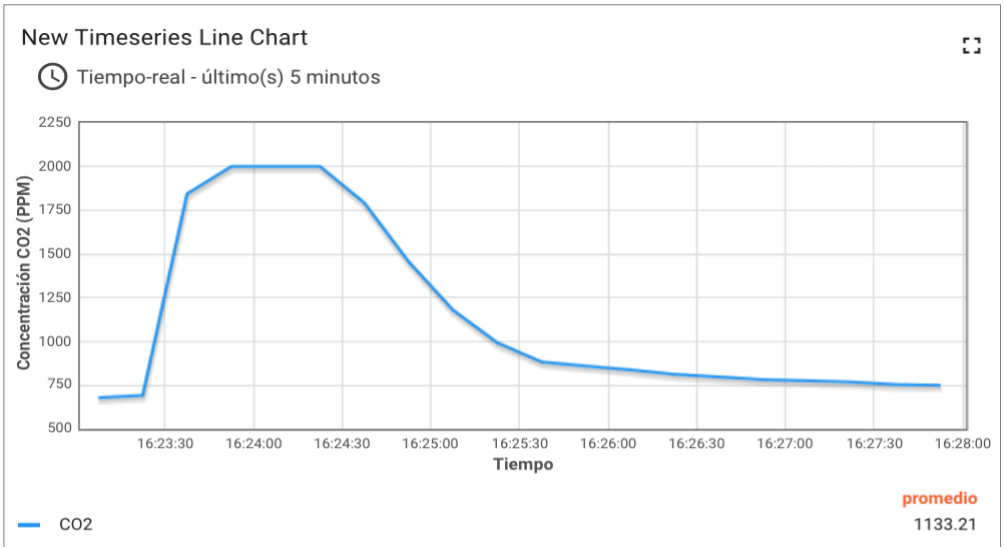

Ilustración 12: Respuesta a cambios bruscos de concentración

Como puede apreciarse, el sensor detecta rápidamente el aumento repentino de concentración de CO2, creciendo esta casi 1.400 PPM en 30 segundos, pero tarda casi 3 minutos en alcanzar de nuevo valores normales, siendo el descenso mucho más gradual. Esto se debe a que, por la misma forma del sensor, la ventilación natural en el interior de este es lenta al no contar con ventilación forzada.

Los cambios de concentración de CO2 en el ambiente son mucho más graduales en la realidad que en esta última prueba, tal y como se puede ver en la evolución temporal de esta a lo largo de 12 horas. Por lo tanto, aunque el sensor no sea capaz de detectar las variaciones más repentinas con total precisión, no tendrá ningún problema para medir los niveles de CO2 en condiciones normales, que es el objetivo de este proyecto.

Basándose en las observaciones realizadas, se concluye que el sensor utilizado para este dispositivo da resultados fieles a la realidad y suficientemente precisos para el uso que se le pretende dar.

4. Diseño final

A estas alturas del proyecto, se ha conseguido crear un sistema capaz de llevar a cabo las funciones básicas que debe cumplir un medidor de CO2, es decir, tomar medidas precisas en tiempo real que puedan visualizarse tanto en una pantalla que forme parte del dispositivo como en una plataforma accesible desde cualquier punto vía Wifi.

El último paso para obtener un producto completo es concretar el diseño de este para que pueda ser utilizado por cualquier usuario. En este apartado, se especificará el funcionamiento definitivo del dispositivo, y se detallarán los últimos aspectos a diseñar para obtener un producto terminado que cumpla los objetivos establecidos al inicio del proyecto.

4.1. Funcionamiento

Al diseñar el prototipo, se han definido qué tareas es capaz de llevar a cabo el dispositivo y cómo las realiza. En este apartado, se especificará en qué momento y cómo llevará a cabo cada tarea, según el uso que se le quiera dar al medidor, para que este lleve a cabo todas las funciones que se marcaron como objetivos.

Las operaciones más importantes que realizará este dispositivo son la medición de datos en tiempo real y el registro de estos en la plataforma Thingsboard, para poder ver así su evolución en el tiempo. Para que esto se cumpla, ambas operaciones deberán llevarse a cabo continuamente. Así, estas tareas deberán repetirse con la frecuencia adecuada sin necesidad de ser activadas de forma externa.

En cuanto a la pantalla OLED del dispositivo, en apartados anteriores esta mostraba constantemente las mediciones de concentración de CO2 y temperatura, las cuales se actualizaban automáticamente cada cinco segundos. Sin embargo, un uso prolongado de la pantalla puede acortar rápidamente la vida útil de esta. La solución a este problema es añadir un sistema de encendido de la pantalla, accionado por el usuario por medio de un pulsador, y un apagado automático al pasar un tiempo determinado. Así, se pueden visualizar las mediciones en cualquier momento, y se alarga el funcionamiento de la pantalla al no estar encendida si no es necesario.

Uno de los objetivos de diseño que se marcaron al inicio del proyecto es crear un sistema de alarma, de modo que este se active cuando el nivel de CO2 medido supere un valor límite predeterminado. El sistema de alarma más adecuado para un dispositivo como este es una señal visual, como podría ser un LED de color rojo. Este LED permanecerá apagado mientras no se supere el nivel de alarma, y se encenderá si se supera.

El valor de este límite que hace que se encienda el LED debería ser el máximo valor permitido en espacios públicos cerrados, el cual puede variar según la normativa, pero se aproximará a las 800 PPM. Sin embargo, un sistema que permitiera establecer este valor en función de las preferencias del usuario haría el uso del medidor mucho más flexible al uso que se le quiera dar en cada situación. Así, de nuevo por medio de uno o varios pulsadores, se añadirá un método para subir o bajar el nivel de alarma.

En cuanto a la calibración de las mediciones, ya se han explicado las diferentes formas de las que esta se puede llevar a cabo, y sus ventajas y desventajas según el uso que se le vaya a dar al dispositivo. Al igual que con la personalización del sistema de alarma, la posibilidad de elegir entre calibración automática o manual según el uso que se le quiera dar al dispositivo sería una gran mejora en el dispositivo. Un ejemplo de esta personalización sería que el medidor utilizara la calibración automática por defecto, pero si en un momento determinado el usuario quisiera iniciar la calibración manual, pueda hacerlo por medio de un pulsador.

Teniendo en cuenta todo lo anterior, podemos definir el funcionamiento del dispositivo. La **medición y el envío de datos** se llevarán a cabo automáticamente, y se repetirán continuamente con una pausa de cinco segundos entre una medición y la siguiente. El **sistema de alarma** estará también en funcionamiento constante comparando cada medida con el valor límite. La calibración será automática hasta que se indique lo contrario. Además, en cualquier momento se pueden activar, por medio de pulsadores, las tres funciones explicadas con anterioridad: **encendido de pantalla**, **cambio del valor de alarma** e inicio de la **calibración manual**.

Para diferenciar cuál de las tres funciones llevar a cabo, lo más sencillo sería instalar tres pulsadores, de modo que cada uno activara una acción diferente. Sin embargo, esta elección también se puede llevar a cabo con un único botón, diferenciando simplemente el periodo que este permanece pulsado. Al contar solo con tres acciones diferentes, las instrucciones a seguir no serían demasiado complejas. Un posible protocolo de empleo se explica a continuación.

- Si se pulsa el botón rápidamente (una duración de pulsado **inferior a 2 segundos**, por ejemplo) se enciende la pantalla, mostrando las mediciones de concentración de CO2 y temperatura. Transcurridos unos segundos, la pantalla se vuelve a apagar.

- Si se pulsa el botón durante un periodo muy largo (por ejemplo, un tiempo **superior a los 5 segundos**) se inicia la calibración manual. Durante los 20 minutos de espera, aparece por pantalla una indicación de que el sistema se está calibrando. Transcurrido este tiempo, se toma una nueva referencia y se apaga la pantalla, volviendo al funcionamiento normal.

- Si el botón se pulsa un tiempo intermedio (**entre 2 y 5 segundos**) se enciende la pantalla, mostrando el valor de alarma actual y pidiendo al usuario que introduzca su nuevo valor. Por medio de activaciones breves del pulsador, el usuario puede cambiar el valor de la alarma. Para salir de este sistema, basta con pulsar el botón de forma prolongada, y la pantalla se apagará.

A continuación, aparece un diagrama de bloques que representa el funcionamiento definido en este apartado.

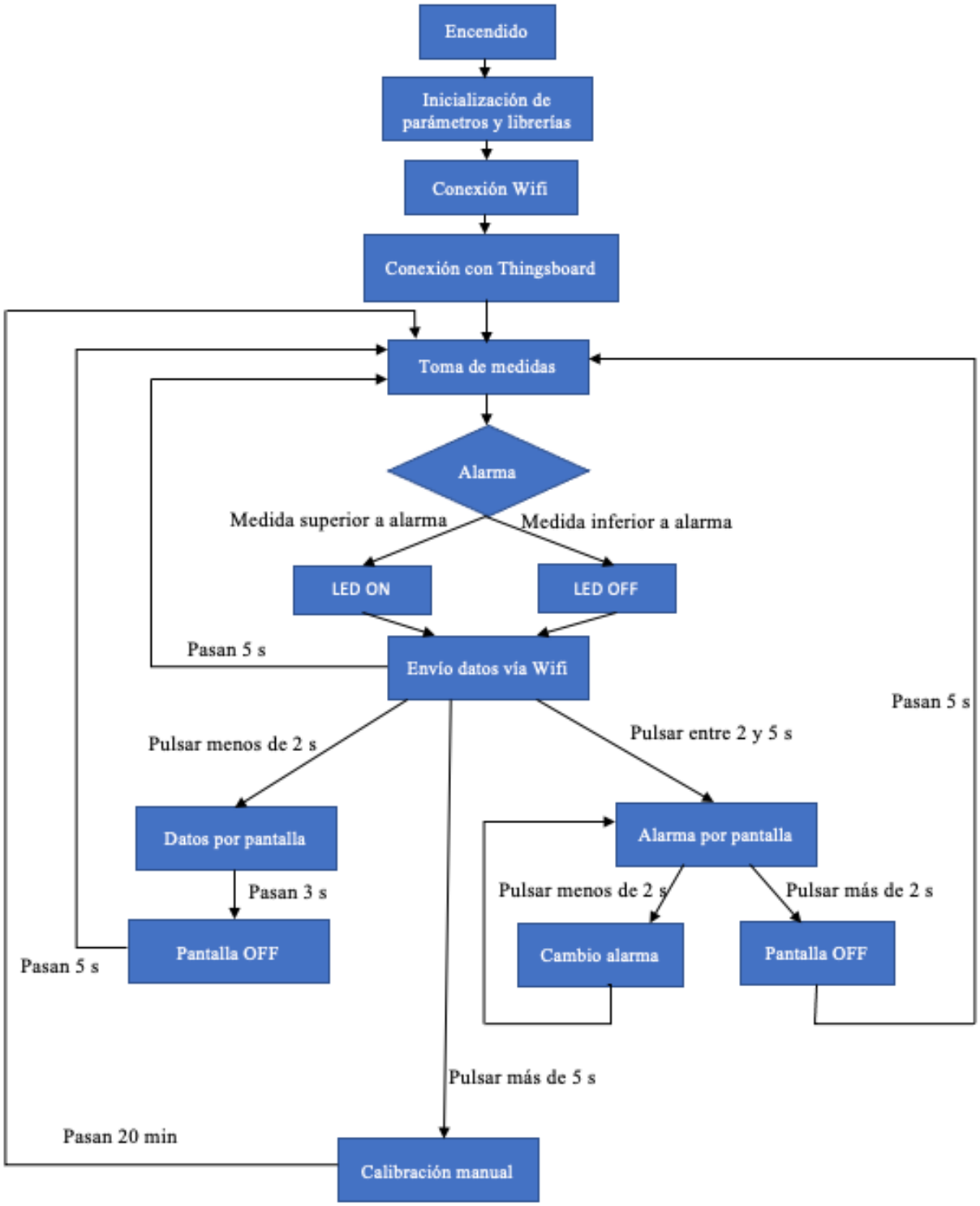

Ilustración 13: Diagrama de flujo del sistema

4.2. Código definitivo

Una vez determinado el sistema operativo de nuestro medidor, es necesario escribir el código que lo haga funcionar correctamente. Este código final se basará en los escritos en apartados anteriores, añadiendo los elementos necesarios para conseguir el funcionamiento deseado. En este apartado se explicará paso a paso el desarrollo de este.

Como ya se ha explicado, las acciones de toma de medidas y envío de datos vía Wifi se realizarán constantemente de forma automática. Para que el tiempo entre medida y medida sea de cinco segundos, se mantiene el sistema de apartados anteriores. Así, se establece un contador después de tomar y enviar cada grupo de medidas, de forma que cuando este alcance los cinco segundos, se repitan las operaciones.

Para añadir el resto de operaciones, es necesario instalar dos nuevos componentes electrónicos al sistema: un **LED** de color rojo y un **pulsador**. Antes de especificar el código que los controla, se explicará su conexión con el microcontrolador.

El LED se deberá conectar por su ánodo con un pin GPIO del sistema, por ejemplo, el **pin D6**. Este pin variará su tensión, según se quiera apagar o encender el LED, entre los 0 y los 5 V, al declararlo OUTPUT en el código. Entre el cátodo y la masa del NodeMCU, debemos añadir una resistencia para que la corriente que atraviesa el LED no supere la máxima permitida, además de limitar al mismo tiempo la intensidad que entrega el microcontrolador. El valor de esta resistencia se calcula en función del voltaje de alimentación y las características del LED, de la siguiente forma:

$$Resistencia\ (Ohmios) = \frac{Voltaje\ de\ alimentación\ (V) - Voltaje\ en\ el\ LED}{Corriente\ en\ el\ LED\ (A)}$$

Siendo el voltaje de alimentación 5 V, la caída de tensión en el LED 2 V y la corriente máxima que lo atraviesa 20 mA, se calcula un valor mínimo de resistencia de 200 Ohmios. Para asegurar tanto el LED como el microcontrolador, se utilizará una resistencia superior, con valor de **470 Ohmios**.

A la hora de incluir el sistema de alarma, basta con añadir una comparación entre cada resultado de la medición y el valor de la alarma, dentro del bucle que se repite cada cinco segundos. Utilizando el comando "digitalWrite()" sobre el pin D6, se consigue que el LED se encienda si el valor medido supera el de la alarma, y se apague si este es inferior. El límite se declarará al inicio del programa y se le otorgará un valor inicial de 800 PPM.

```
1.  //LED encendido si se supera la alarma
2.  if (CO2 >= alarma) {
3.    digitalWrite(LED, HIGH);
4.  }
5.  //Apagado si no se supera
6.  else {
7.    digitalWrite(LED, LOW);
8.  }
```

Para implementar el pulsador, se empleará una conexión **pull-up** con otro de los pines GPIO del microcontrolador, por ejemplo, el **pin D3**. Para realizar esta configuración, se conecta una resistencia al pin de alimentación a 5 V, que a su vez irá conectada por su otro extremo al pulsador, el cual se conectará a la masa del NodeMCU por su otro pin. Por último, se debe conectar el pin elegido, D3, entre la resistencia y el pulsador, obteniendo de este punto el valor de tensión (Vout) que variará según la posición del pulsador. El valor de la resistencia debe ser lo suficientemente elevado para limitar la corriente que sale del microcontrolador, utilizando en este caso una resistencia de **10.000 Ohmios**. El esquema de la conexión pull-up se ve a continuación.

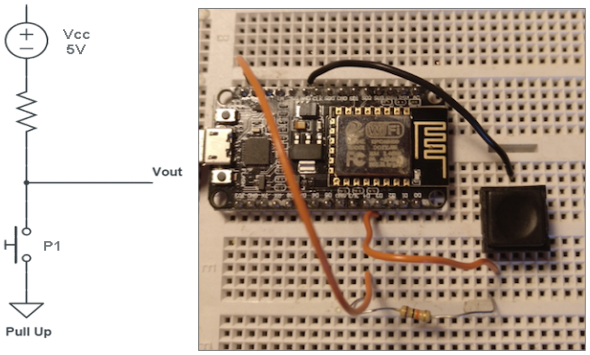

Ilustración 14: Conexión pull-up [14]

Para leer el voltaje que se obtiene en el pin D3, es necesario declarar este como INPUT. El funcionamiento de la configuración pull-up es el siguiente. Cuando el pulsador no cierra el circuito, la tensión leída en D3 es la de la alimentación, es decir 5 V, lo que correspondería a un HIGH. En cuanto se activa el pulsador, se cierra el circuito, cayendo toda la tensión en la resistencia, de modo que D3 tiene un voltaje de 0 V, equivalente a un LOW. Estos dos estados son los que se utilizarán para detectar la activación del pulsador, al poder comprobar el voltaje de D3 con el comando "digitalRead()".

Para detectar los cambios en el pulsador, se crean dos variables, "currentState", el estado del pulsador en cada momento, y "lastState", el estado de este en el instante inmediatamente anterior. Se otorga de inicio un valor HIGH a "lastState", ya que el botón empieza sin estar pulsado, y se actualizan constantemente el valor de "currentState" hasta que se detecta un LOW. Este es el momento en el que el pulsador ha sido activado, al que se define con la variable "pressedTime". Al darse esta situación, cambia el valor de "lastState" a LOW y se actualiza de nuevo el valor de "currentState" hasta detectar un HIGH, que indica que el botón se ha soltado. Este instante se registrará como "releasedTime". Repitiendo este ciclo, es posible detectar cada activación del pulsador, y saber el momento exacto del inicio y el final del pulsado.

Para calcular la duración del pulsado, basta con restar el tiempo de soltado con el tiempo de pulsado. A continuación, aparece el código que detecta la activación del pulsador y mide la duración de esta.

```
1.  //Leer el estado del botón
2.  currentState = digitalRead(BUTTON_PIN);
3.  //Detección de pulsado de botón
4.  if((lastState == 1) && (currentState == 0)) {
5.    pressedTime = millis();
6.    lastState = currentState;
7.  }
8.  //Detteción de soltado de botón
9.  else if((lastState == 0) && (currentState == 1)) {
10.   releasedTime = millis();
11.   lastState = currentState;
12. }
13. //Cálculo del tiempo de pulsado
14. pressDuration = releasedTime - pressedTime;
```

Una vez diseñado el sistema que detecta la duración de los pulsados, basta con compararla con los tiempos que separan las diferentes funciones activadas por el pulsador. Así, si el pulsado es superior a cinco segundos, se llevará a cabo la calibración, si se encuentra entre los dos y los cinco segundos, se entrará en el sistema de cambio de alarma, y si es inferior a dos segundos, se verán los resultados por pantalla.

Un aspecto a tener en cuenta para el correcto funcionamiento de este sistema, debido a la propia estructura del pulsador, es el **efecto de rebote** cuando este se activa o desactiva. Esto significa que, cuando este se pulsa, la tensión medida fluctuará varias veces entre 0 y 5 V en muy poco tiempo, antes de alcanzar un valor estable. Para no detectar estos pulsados no deseados, es necesario asegurar que la duración del pulsado es superior a un valor que haga de **filtro** antes de realizar ninguna función. Este tiempo para evitar los rebotes debe ser suficiente para que el voltaje se haya estabilizado, pero lo más corto posible para no filtrar activaciones reales del pulsador. En este caso, un tiempo de **100 milisegundos** es suficiente, por lo que se añadirá la condición de que el pulsado sea más largo que este valor a las tres comparaciones a realizar.

A continuación, se detalla la programación de las tres funciones manuales que el medidor llevará a cabo, según la duración de la activación del pulsador.

Visualización de datos por pantalla

Cuando se cumple que la duración detectada inferior a dos segundos, pero superior a los 100 ms de filtro, se encenderá la pantalla, mostrando los valores en ese momento de concentración de CO2 y temperatura. Tras tres segundos, la pantalla se apagará automáticamente, y continúa el funcionamiento normal del dispositivo. El código para llevar a cabo esta operación aparece a continuación.

```
1  //Pulsar menos de 2 segundos: encender pantalla
2. if ((pressDuration >= REBOUND_TIME) && (pressDuration < SHORT_PRESS_TIME)&&(modo==0))
3. {
4.    u8g2.clearBuffer();
5.    u8g2_prepare();
6.    u8g2.drawStr( 5, 20,"CO2 (PPM)= ");
7.    u8g2.drawStr( 5+65, 20, CO2str.c_str());
8.    u8g2.drawStr( 5, 20+20, "Temp (C)= ");
9.    u8g2.drawStr( 5+65, 20+20, Tempstr.c_str());
10.   u8g2.sendBuffer();
11.   //Se apaga tras 3 segundos
12.   delay(3000);
13.   u8g2.clearBuffer();
14.   u8g2.sendBuffer();
15.   pressDuration = 0;
16.   pressedTime = 0;
17.   releasedTime = 0;
18. }
```

Calibración manual

Cuando se cumpla la comparativa de duración más larga, se iniciará el proceso de calibración manual. Para ello, se crea una función que se activará cuando el pulsado detectado supere los cinco segundos. En primer lugar, se inicia una espera de 20 minutos mientras por pantalla aparece el mensaje “Calibrando…”. Transcurrido este tiempo, se toma el nuevo valor de referencia, se apaga la pantalla y se vuelve al funcionamiento normal del sistema. De nuevo, aparece esta sección del código a continuación.

```
1.    void calibracionmanual() {
2.      //Se desactiva la calibración automática
3.      myMHZ19.autoCalibration(false);
4.      //Aviso por pantalla
5.      u8g2_prepare();
6.      u8g2.clearBuffer();
7.      u8g2.drawStr( 5, 30,"Calibrando...");
8.      u8g2.sendBuffer();
9.      //20 minutos de espera
10.     delay(12e5);
11.     //Se toma el valor de referencia
12.     myMHZ19.calibrate();
13.     //Se apaga la pantalla
14.     u8g2.clearBuffer();
15.     u8g2.sendBuffer();
16.   }
```

Sistema de cambio de alarma

En el último y más complejo de los casos, cuando la duración del pulsado se encuentre entre los dos y los cinco segundos, el sistema entra en el modo de cambio de alarma. Dentro de este modo, se enciende la pantalla, mostrando el valor actual de la alarma, y se vuelve a detectar pulsados y medir su duración.

Si la duración detectada es inferior a dos segundos, el valor de la alarma aumentará en 100 PPM, apareciendo el nuevo valor por pantalla. Esta operación se repetirá con cada activación corta del pulsador, hasta que la alarma alcance las 2.000 PPM, punto en el cual un nuevo pulsado la reiniciará a 400 PPM. El método para salir del modo de cambio de alarma consiste en mantener pulsado el botón, por lo que cuando se detecte un pulsado superior a dos segundos, se apagará la pantalla y se saldrá de este bucle.

Para entrar y salir del modo de cambio de alarma, se crea una variable, de nombre "modo", que hará la función de flag en el código. Esta variable tendrá inicialmente valor 0, lo cual será condición para entrar en cualquiera de las tres operaciones principales activadas por pulsador. Sin embargo, cuando se entre en el cambio de alarma, el valor de "modo" pasa a ser 1, lo cual solo cumple las condiciones que forman parte de este bucle. Así, mientras "modo" sea 1, solo se podrá modificar la alarma las veces que se quiera, o terminar este proceso de cambio, pero no visualizar medidas por pantalla ni iniciar la calibración manual. Una vez se encuentra un pulsado largo, "modo" vuelve a ser 0 y se resume el funcionamiento normal del dispositivo.

Es importante, para evitar errores y atascos del código en algún bucle, reiniciar los valores de "pressedTime", "releasedTime" y "pressDuration" cada vez que se detecte un pulsado, se calcule su duración y se lleve a cabo la función que le corresponda.

Debido a la extensión de esta parte del código, esta puede encontrarse, junto con el resto del código completo, presentado en el anexo 1. A continuación aparece un esquema representando las conexiones entre los componentes, descritas hasta ahora.

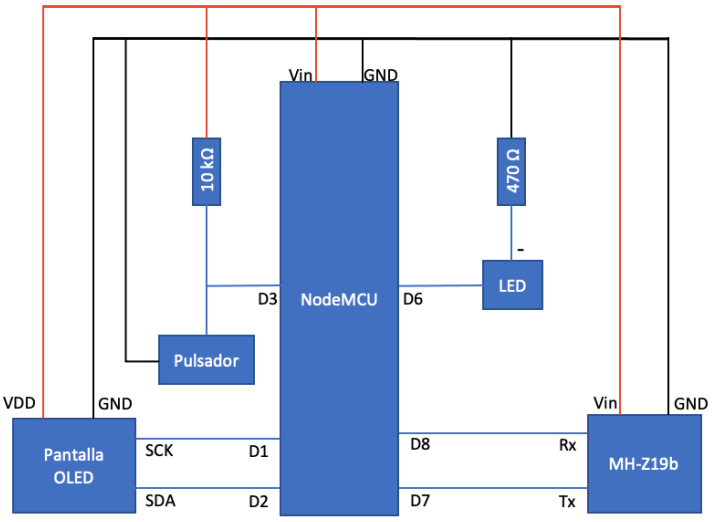

Ilustración 15: Conexiones entre componentes

4.3. Alimentación del sistema

A la hora de elegir una fuente de alimentación para el dispositivo, es fundamental tener en cuenta el uso que se le quiere dar a este. Si se pretende que el medidor esté funcionando continuamente en una misma ubicación durante largos periodos de tiempo, la opción correcta sería alimentar el sistema directamente de una toma de corriente para no interrumpir el funcionamiento al tener que cargar baterías. Si por el contrario se pretende utilizar durante periodo reducido de tiempo, o se quiere desplazar continuamente para medir datos en diferentes espacios, se debería instalar un sistema de baterías recargables con una autonomía suficiente para el uso continuo que se pretenda conseguir.

Al buscar una flexibilidad de uso para este dispositivo, la opción ideal sería un sistema de baterías de gran autonomía, y que pudieran recargarse durante su uso sin interrumpir el funcionamiento del sistema, de forma que el medidor pueda dejarse conectado a una toma de corriente para su uso durante largos periodos de tiempo o desconectarse durante una duración determinada para su uso en diferentes ubicaciones.
Existen muchos tipos diferentes de baterías, dependiendo de los elementos utilizados en su fabricación y funcionamiento. Sin embargo, si se quiere simplificar la conexión de esta con la placa NodeMCU, la solución más sencilla es utilizar un **Power Bank**. Este tipo de baterías recargables, normalmente destinadas a la carga portátil de móviles, pero válidas para cualquier sistema de necesidades similares, utilizan un puerto USB para conectarse al dispositivo a cargar. Además, la tensión de salida de estas baterías es de 5V, precisamente la que necesita el microcontrolador.

A la hora de escoger la batería más adecuada para el dispositivo hay que tener en cuenta, a parte del precio, dos factores fundamentales: su capacidad y sus dimensiones. Para estimar la capacidad que necesita la batería, es necesario calcular el consumo del dispositivo en condiciones normales de funcionamiento. Para ello, se estudiará el consumo de cada uno de los componentes que lo forman.

Placa NodeMCU

El consumo del microcontrolador en condiciones normales de funcionamiento es de 12 mA alimentado a 5 V. Además, en este sistema el microcontrolador transmite repetidamente datos vía Wifi, acción que conlleva un consumo de 120 mA. Sin embargo, el envío de datos se lleva a cabo una vez cada cinco segundos, y la duración de esta operación es muy breve. Si se quisiera ser exactos, se debería calcular cuánto tarda el sistema en enviar cada grupo de datos, para saber qué porcentaje del tiempo se trabaja con el consumo máximo. [15]

Modificando ligeramente el código, es posible medir los instantes exactos en los que se inicia y se finaliza la tarea de envío de resultados, para calcular así su duración. Así, se obtiene un tiempo de envío de unos 2 ms, equivalente a un 0,04% del tiempo total. Este porcentaje es tan bajo que el consumo derivado del envío de datos vía Wifi es despreciable, por lo que se asumirá que el consumo total son los anteriores 12 mA.

```
Tiempo de envío: 1
Tiempo de envío: 1
Tiempo de envío: 2
Tiempo de envío: 2
Tiempo de envío: 2
Tiempo de envío: 2
Tiempo de envío: 2
Tiempo de envío: 2
Tiempo de envío: 2
Tiempo de envío: 2
```

Ilustración 16: Tiempos de envío de datos

$$P = V\ x\ I = 5\ V\ x\ 12x10^{-3}\ A = 0{,}06\ W = 60\ mW$$

Sensor MH-Z19b

Según la hoja de datos del sensor, este consume de media una corriente de 60 mA a una tensión de 5 V. A partir de estos datos, se obtiene directamente el consumo de este componente.

$$P = V\ x\ I = 5\ V\ x\ 60x10^{-3}\ A = 0{,}3\ W = 300\ mW$$

Pantalla OLED

Cuando la pantalla se encuentra encendida, esta consume 11 mA a un voltaje de 3,3 V. Sin embargo, la pantalla permanece apagada la mayor parte del tiempo, por lo que existen dos opciones, estimar el porcentaje de tiempo que la pantalla está encendida, o calcular un valor máximo suponiendo que la pantalla permanece encendida todo el tiempo.

Para obtener una estimación más conservadora, se supondrá esta última suposición, por lo que el consumo de la pantalla será el siguiente.

$$P = V\ x\ I = 3{,}3\ V\ x\ 11x10^{-3}\ A = 0{,}0363\ W = 36{,}3\ mW$$

LED rojo

La caída de tensión en el LED es constante y equivalente a 2 V. Para calcular la intensidad que lo atraviesa hay que tener en cuenta la resistencia a la que va conectado:

$$I = \frac{V_{alimentación} - V_{LED}}{R} = \frac{5\ V - 2\ V}{470\ \Omega} = 0{,}00638\ A = 6{,}38\ mA$$

Así, se obtiene un consumo de 6,38 mA a 2 V. Del mismo modo que la pantalla, el LED no permanecerá encendido constantemente, pero se calcula el consumo máximo que sería necesario en caso de que sí funcionara sin pausa.

$$P = V\ x\ I = 2\ V\ x\ 6{,}38x10^{-3} A = 0{,}013\ W = 13\ mW$$

Resistencias

El último elemento a tener en cuenta son las resistencias que acompañan al LED y el pulsador. En el caso del LED, esta trabaja a 3 V con una intensidad de 6,38 mA, consumiendo así:

$$P = V\ x\ I = 3\ V\ x\ 6{,}38x10^{-3}A = 0{,}019\ W = 19\ mW$$

La resistencia conectada al pulsador solo será atravesada por corriente mientras este se encuentre accionado, lo cual ocurre una pequeña parte del tiempo. Durante este tiempo, el consumo de la resistencia se calcula:

$$P = \frac{V^2}{R} = \frac{(5\ V)^2}{10000\ \Omega} = 0{,}0025\ W = 2{,}5\ mW$$

Teniendo en cuenta lo bajo que es este valor y el poco tiempo que el pulsador permanece activado durante el uso normal del dispositivo, se puede suponer que el consumo de esta última resistencia es nulo.

Una vez calculados los consumos de todos los componentes del sistema, se obtiene un consumo total de **428,3 mW**, equivalente, referido a un voltaje de alimentación de 5 V, a una intensidad de aproximadamente 86 mA. Si se quiere un medidor con una autonomía de un mínimo de 24 horas, la capacidad de la batería a instalar deberá ser superior a 2064 mAh. De esta forma, se procederá a elegir una batería óptima por dimensiones y precio que ofrezca como mínimo esta capacidad.

Tras comparar las diferentes Power Bank en el mercado, la batería elegida finalmente es el modelo **Vancely Mini Power Bank**, cuyas características aparecen a continuación. [15]

- Voltaje de salida 5 V.
- Intensidad máxima de salida 2,4 A.
- Capacidad 10.000 mAh.
- Puertos USB, Micro USB.
- Protección a sobrecorriente, sobrevoltaje, sobrecarga y cortocircuito.
- Dimensiones: 9,1 x 6,2 x 2,8 cm.
- Precio: 11 €.

Ilustración 17: Vancely Mini Power Bank

4.4. Diseño de la carcasa y montaje final

El último paso para obtener un producto utilizable directamente por el consumidor es diseñar una carcasa, en cuyo interior se encontrarán los componentes que hasta ahora formaban la totalidad del dispositivo. Para ello, es necesario decidir en primer lugar la disposición fija de estos componentes una vez se lleve a cabo el montaje.

El factor principal a la hora de elegir cómo estarán dispuestos los componentes es buscar un diseño compacto, de forma que quepan en el menor espacio posible. De entre todos ellos, la batería es la que ocupa, con diferencia, un mayor espacio, por lo que será la que afecte más directamente al tamaño final de la carcasa.

Otro aspecto a tener en cuenta es que el pulsador, el LED, la pantalla y el puerto de carga de la batería deben estar expuestas al exterior. Además, las entradas de aire del sensor deben estar lo más expuestas posible para favorecer la medición. Teniendo esto en cuenta, se puede idear una primera disposición de todos los componentes, la cual se ve a continuación.

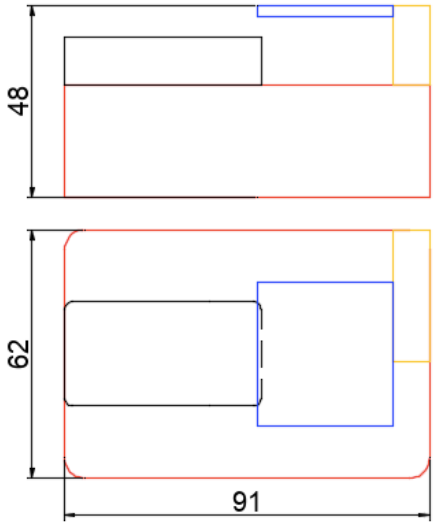

Ilustración 18: Disposición de los componentes en la carcasa

En este esquema aparecen, con sus respectivos tamaños relativos, las piezas de mayor tamaño del sistema: la batería, marcada con color rojo, el microcontrolador, a color negro, la pantalla, de color azul, y el sensor, de color dorado. Las piezas más pequeñas, el pulsador, el LED y las resistencias, no aparecen ya que, debido a su tamaño, no van a limitar en exceso las dimensiones de la carcasa.

Como se puede ver, dos de las dimensiones máximas del conjunto vienen dadas por las de la batería, sobre la cual se montará el microcontrolador, añadiendo una placa de circuito y un zócalo de pines. El sensor y la pantalla irán anclados a las paredes de la carcasa, de forma que puedan tener salida al exterior. Siguiendo esta disposición, las dimensiones mínimas de una carcasa en la que quepan todos los componentes son de **91x62x48 milímetros**.

A la hora de diseñar la carcasa, simplemente se necesitaría fabricar un hexaedro hueco de estas dimensiones, o unas un poco superiores para que el encaje no sea tan justo. A esta forma básica bastaría con añadirle las aperturas necesarias para los componentes que dan al exterior, y alguna vía de ventilación para evitar el sobrecalentamiento del dispositivo.

Los materiales más utilizados para fabricar carcasas de este tipo son el policarbonato (PC) y el acrilonitrilo butadieno estireno (ABS). En cuanto al método de fabricación, la opción más adecuada sería utilizar una impresora 3D para crear la carcasa diseñada con uno de los materiales anteriores.

Sin embargo, principalmente por falta de presupuesto, para este proyecto se va a utilizar una caja ya existente, cuyas características aparecen a continuación.

- Fabricante: Diotronic.
- Material: ABS.
- Dimensiones: 100 x 74 x 50 mm.
- Precio: 8 €.

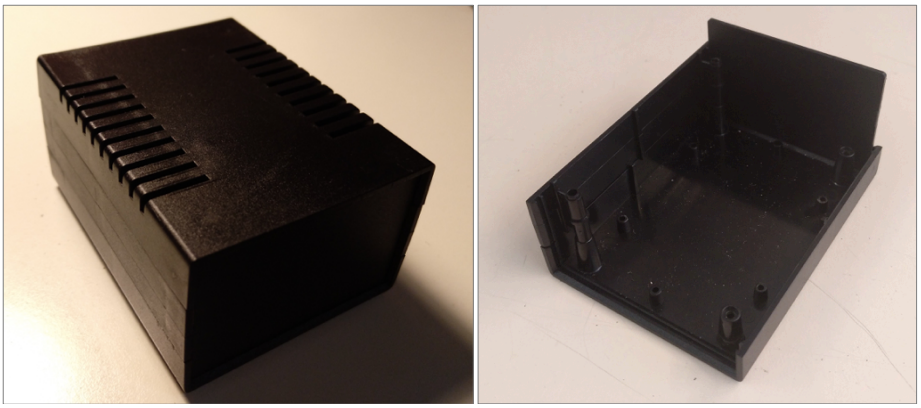

Ilustración 19: Carcasa del medidor

A primera vista, estas dimensiones permiten incluir todos los componentes en su interior. Sin embargo, los tornillos que mantienen la caja cerrada disminuyen el espacio útil en su interior, por lo que la batería es demasiado grande. Así, para este primer modelo de medidor, la batería se encontrará en el exterior de esta carcasa, mientras que el resto de los componentes estarán montados en su interior.

El proceso de montaje es simple. El sensor y el microcontrolador se soldarán a una placa fija en la base de la carcasa sobre la que se realizarán las conexiones necesarias. El resto de los componentes irán anclados a las paredes para estar en contacto con el exterior. Al tener ya esta carcasa ranuras de ventilación, solo es necesario crear nuevas aperturas para estos elementos. Finalmente, la batería quedará fuera de la carcasa, conectada con el microcontrolador a través de una apertura en la carcasa.

5. Resultados

Una vez realizado el montaje final, el resultado de este proyecto es el medidor de CO2 que aparece en la imagen siguiente.

Ilustración 20: Medidor de CO2 completado

Las capacidades y características de este medidor se han explicado en detalle a lo largo del proyecto, pero quedan resumidas a continuación.

- Medición de concentración de CO2 y temperatura en tiempo real.
- Compensación automática del efecto de la temperatura en la medición.
- Rango de medición hasta 2.000 PPM.
- Visualización de datos a través de una pantalla con encendido por pulsador.
- Visualización de resultados y su evolución en una plataforma online.
- Sistema de alarma visual con límite variable.
- Calibración automática o manual según el uso deseado.
- Consumo medio de 430 W.
- Voltaje de alimentación de 5 V.
- Alimentación y carga de la batería por vía microUSB.
- Autonomía de la batería de 96 horas.

El coste final de fabricación, calculado como la suma de los precios de todos los componentes, es de **56 €**, muy inferior a los modelos que se analizaron al inicio del proyecto, incluso los más simples que ofrecían menos servicios de los que es capaz este medidor.

5.1. Limitaciones y posibilidades de mejora

A continuación, se analizan las limitaciones del medidor desarrollado, y se plantean posibles mejoras a implementar si se quisiera continuar el proyecto y producir más dispositivos para su uso extendido.

- El diseño de la carcasa utilizado para el primer modelo transporta y protege los componentes más pequeños del dispositivo, pero el hecho de que la batería quede en el exterior supone un diseño incómodo y poco estético. Si se quisiera continuar con el proyecto, sería fundamental diseñar una carcasa de cero, de tal forma que tanto la batería como el cable que la une con el microcontrolador queden en su interior.

- Los datos de la red Wifi a la que se conecte el medidor se deben establecer antes de cargar el código al microcontrolador, lo que limita mucho la flexibilidad de uso del dispositivo. El hecho de que el puerto USB de la placa NodeMCU esté expuesto facilita esta tarea, pero si se quisiera producir el medidor a gran escala para su uso en espacios variados, la solución será añadir una forma de elegir la red Wifi durante el uso del dispositivo, lo cual supone un gran cambio en el diseño de este.

- El sensor utilizado para el medidor mide y compensa el efecto de la temperatura en la medición de concentración de CO2. Sin embargo, no es capaz de medir la presión atmosférica, por lo que el efecto de esta no puede compensarse. Si el dispositivo se va a utilizar en un mismo entorno, como puede ser una misma ciudad, este efecto no será demasiado importante. Si se quisiera expandir la fabricación y la distribución del medidor, sería interesante añadir un sensor de presión y compensar su efecto en los resultados.

- Para este proyecto se ha utilizado la Demo de Thingsboard, ya que es gratuita y muy útil para la visualización de resultados. Sin embargo, esta versión no permite la exportación de datos en caso de que se quisiera analizar estos en otra plataforma como podría ser Excel. Si esta herramienta se considerara útil en un futuro, se debería plantear la posibilidad de contratar la versión completa de Thingsboard, con un precio de 10 € al mes.

- El uso del medidor no es complejo, pero requiere recordar los diferentes tiempos que se debe activar el pulsador para llevar a cabo diferentes operaciones. Una solución simple sería imprimir o serigrafiar unas instrucciones sencillas en la misma carcasa del dispositivo.

5.2. Conclusiones

A la hora de sacar conclusiones del trabajo realizado, es necesario hacer un balance entre los objetivos marcados al inicio de este y el dispositivo obtenido como resultado final.

El medidor desarrollado cumple todas las operaciones que se consideraban fundamentales para un dispositivo de este tipo. Además, a lo largo del proyecto se han ideado más funciones, especialmente dirigidas a la flexibilidad en el uso del medidor, las cuales se han implementado con éxito en el diseño de este. En relación con las capacidades del dispositivo, se puede considerar que el desarrollo de este ha sido satisfactorio.

Al compararlo con otros medidores, el dispositivo creado supera a alguno de los ya existentes en capacidades, y a todos los productos analizados en precio, por lo que podría competir, a priori, con productos existentes. Sin embargo, como ya se ha explicado con anterioridad, existen mejoras que serían de gran utilidad si se implementaran en el dispositivo en un futuro, especialmente las referidas a la carcasa para dar al dispositivo un aspecto más profesional y un uso más cómodo. Estas mejoras serían necesarias para que el medidor aquí desarrollado pudiera competir realmente con otros. Así, se puede concluir que, aunque el medidor en su estado actual es demasiado básico para ser una opción real, la continuación de su diseño podría conducir a un producto competitivo en el mercado.

Finalmente, en cuanto a los objetivos académicos que aparecieron al principio del proyecto, considero que he adquirido competencias muy útiles durante la realización de este trabajo. Los conocimientos previos sobre electrónica y programación adquiridos durante la carrera has sido de gran utilidad, pero el aplicarlos a un proyecto propio según las necesidades en cada momento hace que se afiancen mucho más. Además, considero que la competencia más fundamental desarrollada a lo largo de este proyecto es la resolución de problemas, es decir, buscar información en diversas fuentes, o probar diferentes opciones, cada vez que aparecía un obstáculo a lo largo del proyecto.

6. Bibliografía

[1] Gobierno de España.
https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/COVID19_Medidas_centros_educativos_Curso_2020_2021.pdf

[2] Medidor de CO2 PCE-AQD 50.
https://www.pce-instruments.com/espanol/instrumento-medida/medidor/medidor-de-co2-pce-instruments-medidor-de-co2-pce-aqd-50-det_5958099.htm

[3] Medidor de CO2 Dioxcare Winix.
https://dortomedical.com/purificadores-de-aire/2213-medidor-de-co2-dioxcare-pdf.html

[4] Medidor CO2 Digital VDL.
https://www.servovendi.com/es/medidor-controlador-de-humedad-temperatura-co2-vdl.html

[5] Medidor CO2 Básico.
https://dortomedical.com/diagnostico/1887-medidor-co2-digital-vdl-.html?utm_source=google&utm_medium=surfaces&utm_campaign=shopping_feed&utm_content=Fichas%20de%20anuncios%20gratuitas%20de%20Google%20Merchant&gclid=Cj0KCQjwna2FBhDPARIsACAEc_Xbdodfyr5J4q_2b7uKNS1RD6qAeXZuGoPYi6xK01Pd5Pqlc3WRJ1UaAtAsEALw_wcB

[6] DescubreArduino sobre la placa NodeMCU.
https://descubrearduino.com/nodemcu/

[7] Medición del CO2 por infrarrojos.
https://www.vaisala.com/sites/default/files/documents/CEN-TIA-Parameter-How-to-measure-CO2-Application-note-B211228ES-A.pdf

[8] Esquema de la detección de partículas por infrarrojos.
http://www.academiatesto.com.ar/cms/absorcion-infrarroja-proceso-ir

[9] Winsen. Sensor MH-Z19b.
https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf

[10] “Jonathan Dempsey”. Librería MH-Z19.
https://github.com/WifWaf/MH-Z19

[11] “oliver”. Librería u8g2.
https://github.com/olikraus/u8g2

[12] Plataforma Thingsboard.
https://thingsboard.io

[13] “ThingsBoard Team”. Librería Thingsboard.
https://github.com/thingsboard/ThingsBoard-Arduino-MQTT-SDK

[14] Esquema conexión pull-up.
https://programarfacil.com/blog/arduino-blog/resistencia-pull-up-y-pull-down/

[15] Datasheet del chip ESP8266.
https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[16] Vancely Mini Power Bank.
https://www.amazon.es/Vancely-10000Mah-Cargador-Capacidad-Smartphones/dp/B07TSCTG2F/ref=sr_1_5?dchild=1&keywords=power+bank&qid=1623868631&sr=8-5

7. Apéndices

Anexo 1: Código completo

```
1.  //Librerías a utilizar
2.  #include <Arduino.h>
3.  #include "MHZ19.h"
4.  #include <SoftwareSerial.h>
5.  #include <U8g2lib.h>
6.  #include <Wire.h>
7.  #include "ThingsBoard.h"
8.  #include <ESP8266WiFi.h>
9.  #include <ArduinoJson.h>
10. #include <PubSubClient.h>

11. //Objeto para controlar la pantalla
12. U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE,D1,D2);

13. //Pines de conexión con el sensor
14. #define RX_PIN D7
15. #define TX_PIN D8
16. //Pines para el pulsador y el LED
17. const int LED = D6;
18. const int BUTTON_PIN = D3;

19. //Velocidad de comunicación en serie
20. #define BAUDRATE 9600
21. //Tiempo filtro para evitar el rebote
22. const int REBOUND_TIME = 100;
23. //Tiempos de pulsado para diferentes operaciones
24. const int SHORT_PRESS_TIME = 2000;
25. const int LONG_PRESS_TIME = 5000;
26. //Variables para las mediciones
27. int CO2;
28. int Temp;
29. String CO2str;
30. String Tempstr;
31. //Variable para el contador
32. unsigned long getDataTimer = 0;
33. //Estados del pin del pulsador
34. int lastState = 1;
35. int currentState;
36. //Variables para calcular la duración de pulsado
37. unsigned long pressedTime  = 0;
38. unsigned long releasedTime = 0;
39. long pressDuration = 0;
40. //Valor variable de la alarma
41. int alarma = 800;
42. String alarmastr;
43. //Variable flag para entrar y salir del cambio de alarma
44. int modo = 0;
```

```
//Serial de comunicación con el sensor
MHZ19 myMHZ19;
SoftwareSerial mySerial(RX_PIN, TX_PIN);

//Datos de la conexión Wifi
#define WIFI_AP           "MOVISTAR_2E99"
#define WIFI_PASSWORD     "3ft35u2q27EhcumQLnPw"
//#define WIFI_AP           "Aquaris X"
//#define WIFI_PASSWORD     "123456ABC"
//Datos del dispositivo Thingsboard
#define TOKEN             "v42uu5eMqsqljx6gwyRC"
#define THINGSBOARD_SERVER  "demo.thingsboard.io"
WiFiClient espClient;
ThingsBoard tb(espClient);
int status = WL_IDLE_STATUS;

void setup() {
  //Inicio del serial
  mySerial.begin(BAUDRATE);
  myMHZ19.begin(mySerial);
  //Inicio de la pantalla
  u8g2.begin();

  //Calibración automática por defecto
  myMHZ19.autoCalibration(true);
  //Cambio del rango de medida
  myMHZ19.setRange(2000);

  //Conexión a la red Wifi
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();

  //Configuración de los pines
  pinMode(LED, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}

void loop() {
  //Reconexión con la red Wifi
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }
  //Conexión con Thingsboard
  if (!tb.connected()) {
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      return;
    }
  }
  //Contador para repetición de las operaciones principales
  if (millis() - getDataTimer >= 5000) {
    //Toma de medidas
    CO2 = myMHZ19.getCO2();
    Temp = myMHZ19.getTemperature();
    CO2str = String(CO2);
   Tempstr = String(Temp);
```

```
94.     //LED encendido si se supera la alarma
95.     if (CO2 >= alarma) {
96.       digitalWrite(LED, HIGH);
97.     }
98.     //LED apagado si no se supera
99.     else {
100.     digitalWrite(LED, LOW);
101.    }

102.    //Envío de datos a Thingsboard
103.    tb.sendTelemetryString("CO2", CO2str.c_str());
104.    tb.sendTelemetryString("Temperature", Tempstr.c_str());
105.    tb.loop();
106.    getDataTimer = millis();
107.  }

108.  //Leer el estado del pulsador
109.  currentState = digitalRead(BUTTON_PIN);

110.  //Detección de activación del pulsador
111.  if((lastState == 1) && (currentState == 0)) {
112.    pressedTime = millis();
113.    lastState = currentState;
114.  }
115.  //Detteción de soltado del pulsador
116.  else if((lastState == 0) && (currentState == 1)) {
117.    releasedTime = millis();
118.    lastState = currentState;
119.  }
120.  //Cálculo de la duración de pulsado
121.  pressDuration = releasedTime - pressedTime;

122.  //Pulsar más de 5 segundos: calibrar
123.  if ((pressDuration >= LONG_PRESS_TIME)&&(modo == 0)) {
124.    calibracionmanual();
125.    //Reinicio de variables
126.    pressDuration = 0;
127.    pressedTime = 0;
128.    releasedTime = 0;
129.  }

130.  //Pulsar entre 2 y 5 segundos: cambio de alarma
131.  if (((pressDuration >= SHORT_PRESS_TIME) && (pressDuration <
      LONG_PRESS_TIME))&&(modo == 0)) {
132.    //Cambio de la variable flag
133.    modo = 1;
134.    //Valor de alarma por pantalla
135.    alarmastr = String(alarma);
136.    u8g2_prepare();
137.    u8g2.clearBuffer();
138.    u8g2.drawStr(5, 20, "Nuevo valor de alarma:");
139.    u8g2.drawStr(5, 40, alarmastr.c_str());
140.    u8g2.sendBuffer();
141.    //Reinicio de variables
142.    pressDuration = 0;
143.    pressedTime = 0;
```

```
144.   releasedTime = 0;
145.  }

146.  //Subir la alarma con pulso corto hasta 2000
147.  if ((pressDuration > REBOUND_TIME) && (pressDuration <= SHORT_PRESS_TIME) && (alarma
      < 2000)&&(modo==1)) {
148.   alarma = alarma + 100;
149.   alarmastr = String(alarma);
150.   u8g2.clearBuffer();
151.   u8g2.drawStr(5, 20, "Nuevo valor de alarma:");
152.   u8g2.drawStr(5, 40, alarmastr.c_str());
153.   u8g2.sendBuffer();
154.   //Reinicio de variables
155.   pressDuration = 0;
156.   pressedTime = 0;
157.   releasedTime = 0;
158.  }

159.  //Vuelta a 400 al llegar a 2000
160.  if ((pressDuration > REBOUND_TIME) && (pressDuration <= SHORT_PRESS_TIME) && (alarma
      == 2000)&&(modo==1)) {
161.   alarma = 400;
162.   alarmastr = String(alarma);
163.   u8g2.clearBuffer();
164.   u8g2.drawStr(5, 20, "Nuevo valor de alarma:");
165.   u8g2.drawStr(5, 40, alarmastr.c_str());
166.   u8g2.sendBuffer();
167.   //Reinicio de variables
168.   pressDuration = 0;
169.   pressedTime = 0;
170.   releasedTime = 0;
171.  }

172.  //Salir de cambio de alarma con pulso largo
173.  If ((pressDuration > SHORT_PRESS_TIME)&&(modo==1)){
174.   //Cambio del flag para salir del cambio de alarma
175.   modo = 0;
176.   //Apagar pantalla
177.   u8g2.clearBuffer();
178.   u8g2.sendBuffer();
179.   //Reinicio de variables
180.   pressDuration = 0;
181.   pressedTime = 0;
182.   releasedTime = 0;
183.  }

184.  //Pulsar menos de 2 segundos: encender pantalla
185.  if ((pressDuration >= REBOUND_TIME) && (pressDuration <
      SHORT_PRESS_TIME)&&(modo==0)) {
186.   u8g2.clearBuffer();
187.   u8g2_prepare();
188.   u8g2.drawStr( 5, 20,"CO2 (PPM)= ");
189.   u8g2.drawStr( 5+65, 20, CO2str.c_str());
190.   u8g2.drawStr( 5, 20+20, "Temp (C)= ");
191.   u8g2.drawStr( 5+65, 20+20, Tempstr.c_str());
192.   u8g2.sendBuffer();
```

```
193.   //Se apaga tras 3 segundos
194.   delay(3000);
195.   u8g2.clearBuffer();
196.   u8g2.sendBuffer();
197.   //Reinicio de variables
198.   pressDuration = 0;
199.   pressedTime = 0;
200.   releasedTime = 0;
201.  }
202.}

203.//Función de preparación de la pantalla
204.void u8g2_prepare(void) {
205.  u8g2.setFont(u8g2_font_6x10_tf);
206.  u8g2.setFontRefHeightExtendedText();
207.  u8g2.setDrawColor(1);
208.  u8g2.setFontPosTop();
209.  u8g2.setFontDirection(0);
210.}

211.//Función de inicio de conexión Wifi
212.void InitWiFi() {
213.  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
214.  while (WiFi.status() != WL_CONNECTED) {
215.   //Aviso por pantalla
216.   u8g2_prepare();
217.   u8g2.clearBuffer();
218.   u8g2_prepare();
219.   u8g2.drawStr( 5, 30,"Conectando...");
220.   u8g2.sendBuffer();
221.   delay(500);
222.  }
223.  //Apagado de pantalla
224.  u8g2.clearBuffer();
225.  u8g2.sendBuffer();
226.}

227.//Función de reconexión
228.void reconnect() {
229.  status = WiFi.status();
230.  if ( status != WL_CONNECTED) {
231.   WiFi.begin(WIFI_AP, WIFI_PASSWORD);
232.   while (WiFi.status() != WL_CONNECTED) {
233.    delay(500);
234.   }
235.  }
236.}

237.//Función de calibración manual
238.void calibracionmanual() {
239.  //Desactivación de calibración automática
240.  myMHZ19.autoCalibration(false);
241.  //Aviso por pantalla
242.  u8g2_prepare();
243.  u8g2.clearBuffer();
244.  u8g2.drawStr( 5, 30,"Calibrando...");
```

```
245.  u8g2.sendBuffer();
246.  //20 minutos de espera
247.  delay(12e5);
248.  //Toma del nuevo valor de referencia
249.  myMHZ19.calibrate();
250.  //Apagado de pantalla
251.  u8g2.clearBuffer();
252.  u8g2.sendBuffer();
253.}
```

Anexo 2: Otras opciones para el sensor

- **AZ-Delivery Sensor de Gas MQ-2:**
  - Voltaje de alimentación: 5 V.
  - Detección de GLP, i-butano, propano, metano, alcohol, hidrógeno y humo.
  - Sensor electrónico-químico.
  - Rango: 100 – 10.000 PPM.
  - Umbral límite ajustable por potenciómetro.
  - Salida analógica para la medición.
  - Salida digital para comparación con el umbral.
  - Precio: 4,99 €.

Enlace para el sensor MQ-2:
https://www.amazon.es/AZDelivery-sensor-Calidad-Módulo-Arduino/dp/B07CYYB82F/ref=sr_1_9?__mk_es_ES=ÅMÅŽÕÑ&crid=2GJF4B8DC6F5F&dchild=1&keywords=sensor%2Bco2%2Barduino&qid=1624296709&sprefix=sensor%2BCO2%2B%2Caps%2C241&sr=8-9&th=1

- Sensor CO2 MH-Z19:
  - Voltaje de alimentación: 5 (± 1) V.
  - Nivel de interfaz: 3,3 V.
  - Corriente media < 40 mA.
  - Corriente máxima: 125 mA.
  - Rango de medición: 400 – 5.000 PPM.
  - Temperatura de funcionamiento: -10 – 50 ºC.
  - Humedad de funcionamiento: 0 – 95 %.
  - Salidas: analógica, PWM y UART.
  - Precio: 26 €.

Enlace para el sensor MH-Z19:
https://store.prometec.net/producto/sensor-co2-mh-z19/

Anexo 3: Otras opciones para la pantalla

- **AZ-Delivery OLED I2C de 0,91 pulgadas:**
  - Voltaje de alimentación: 3,3 / 5 V.
  - Interfaz de comunicación I2C.
  - Tamaño de pantalla: 26 x 9,5 mm.
  - Resolución: 128 x 32 píxeles.
  - Precio: 5,99 €.

Enlace para la pantalla OLED de 0,91 pulgadas:
https://www.amazon.es/AZDelivery-128-160-p%C3%ADxeles-91-OLED/dp/B079H2C7WH/ref=sr_1_6?__mk_es_ES=ÅMÅŽÕÑ&dchild=1&keywords=pantalla+oled+32&qid=1624296493&s=electronics&sr=1-6

- **AZ-Delivery HD44780 LCD:**
  - Voltaje de alimentación: 3,3 / 5 V.
  - Dimensiones: 80 x 36 x 12,5 mm.
  - Resolución: 16 caracteres x 2 líneas.
  - Área de visualización: 12 x 56 mm.
  - Retroiluminación verde.
  - Precio: 5,79 €.

Enlace para la pantalla LCD:
https://www.amazon.es/AZDelivery-HD44780-visualización-16-caracteres-Arduino/dp/B079T264ZZ/ref=sr_1_5?__mk_es_ES=ÅMÅŽÕÑ&crid=1BZ2IZUDIKHJJ&dchild=1&keywords=pantalla+lcd+arduino&qid=1624296088&s=electronics&sprefix=pantalla+lcd+%2Celectronics%2C176&sr=1-5